SVM Based Learning System For Information Extraction

Yaoyong Li, Kalina Bontcheva, and Hamish Cunningham

Department of Computer Science, The University of Sheffield, Sheffield, S1 4DP, UK
{yaoyong,kalina,hamish}@dcs.shef.ac.uk

Abstract. We present an SVM-based learning algorithm for information extraction, including experiments on the influence of different algorithm settings. Our approach needs fewer SVM classifiers to be trained than other recently proposed SVM-based systems. Another distinctive feature is the use of a variant of the SVM, the SVM with uneven margins, which is particularly helpful for mixed-initiative (adaptive) information extraction. We also compare our system to other state of the art systems (including rule learning and statistical learning algorithms) on three IE benchmark datasets: CoNLL-2003, the CMU seminars corpus, and the software jobs corpus. The experimental results showed that our system had a compatible performance. It outperformed a recent SVM system, achieved the highest scores on eight out of 17 categories on the jobs corpus, and was second best on the remaining nine.

1 Introduction

Information Extraction (IE) is the process of automatic extraction of information about pre-specified types of events, entities or relationships from text such as newswire articles or Web pages (see [10] for a comprehensive overview of IE). A lot of work has been done on named entity recognition, a basic task of IE, which aims to classify the proper nouns and/or numerical information in documents. Actually most IE tasks can be viewed as the task of recognising some information entities from the text. IE can be useful in many applications, such as information gathering in a variety of domains, automatic annotations of web pages for semantic web, and knowledge management.

Machine learning techniques have been used for IE and achieved state of the art results. In the applications of machine learning to IE, a learning algorithm usually extracts 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. Usually the algorithm would learn a more accurate model if given more training examples. However, manual annotation is a time-consuming process. Hence, in many applications the so called adaptive or mixed-initiative learning is desirable (see e.g., Alembic [14], Amilcare [9]). In a mixed-initiative IE system, a few documents are manually annotated first (i.e., the user has the initiative to begin with). The system learns an initial model from this small pool of annotated examples. Then the model is applied to tag new documents (the

system starts having some initiative by suggesting the tags) and the results are corrected by the user. Then the system updates the model based on the user's corrections, and the process continues until the user is satisfied with the system performance and allows it to work fully automatically. In order to lower the overhead of training a learning system, this kind of human-machine interactive approach is crucial for building an efficient and flexible IE system. Therefore, an important part of this work is focused on evaluating our learning algorithm on growing amounts of data, starting from a small set of annotated documents.

Machine learning algorithms for IE can be classified broadly into two main categories: rule learning and statistical learning. The former induces a set of rules from a training set, while the later learns a statistical model or classifiers. Support Vector Machines (SVM) is a general supervised machine learning algorithm, that has achieved state of the art performance on many classification tasks, including named entity recognition (see e.g. [21], [24]). [21] compared three commonly used methods for named entity recognition – the SVM with quadratic kernel, maximal entropy method, and a rule based learning system, and showed that the SVM based system performed better than the other two. In our view, the comparison between different learning methods in [21] is more informative than the comparison in, e.g., the CoNLL-2003 share task (see [27]), because the former used both the same corpus and the same features for all the systems, while in the later different systems used the same corpus but different features.[1] [21] also described an efficient implementation of the SVM with quadratic kernel. [24] used a lattice-based approach to named entity recognition and employed the SVM with cubic kernel to compute transition probabilities in a lattice. Their results on CoNLL-2003 shared task were comparable to other systems but were not the best ones.

This paper describes an SVM-based learning algorithm for IE and present detailed experimental results. In contrast to previous similar work, our SVM model (see Section 2) uses an uneven margins parameter which has been shown in [23] to improve the performance for document categorisation (especially for small categories). Detailed experiments to investigate different experimental settings of the SVM based algorithm on several benchmark datasets was carried out (see Section 4.1). The experimental datasets were chosen to enable thorough comparisons between our approach and other state of the art learning algorithms (see Section 4.2). The algorithm was also evaluated in simulated mixed-initiative settings, where only a small number of documents was given for learning initially and then more and more documents were provided incrementally. Section 5 covers related work.

[1] The still ongoing Pascal Challenge in evaluation of machine learning methods for IE aims to provide a corpus and a pre-defined set of features, so different algorithms can be compared better (http://nlp.shef.ac.uk/pascal/).

2 The SVM Based Learning Algorithm

We used a variant of the SVM, the SVM with uneven margins [23], which has a better generalisation performance than the original SVM on datasets where the positive examples are much less than the negative ones. The uneven marginal parameter has been shown previously to facilitate document classification on unbalanced training data (see [23]). Given that IE classification tasks, particularly when learning from small data sets, often involve unbalanced data, we decided to use SVM with uneven margins, instead of the original SVM algorithm.

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:

$$\begin{aligned} \text{minimise}_{\mathbf{w},\ b,\ \xi}\ & \langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{i=1}^{m} \xi_i \\ \text{subject to} \quad & \langle \mathbf{w}, \mathbf{x}_i \rangle + \xi_i + b \geq 1 \qquad \text{if} \ \ y_i = +1 \\ & \langle \mathbf{w}, \mathbf{x}_i \rangle - \xi_i + b \leq -\tau \quad \text{if} \ \ y_i = -1 \\ & \xi_i \geq 0 \qquad\quad \text{for} \ \ i = 1, ..., m \end{aligned}$$

In these equations, τ is the uneven marginal parameter which is the ratio of negative margin to the positive margin in the classifier and is equal to 1 in the original SVM. [23] also showed that the solution of the above problem could be obtained by solving a related SVM problem.

2.1 Use of Context in the Feature Vectors

When statistical learning methods are applied to IE tasks, they are typically formulated as classification, i.e., each word in the document is classified as belonging or not to one of the target classes (e.g., named entity tags). The same strategy was adopted in this work, which effectively means that each word is regarded as a separate instance by the SVM classifier. First an input vector is formed, based on a large number of features. Since in IE the context of the word is usually as important as the word itself, the features in the input vector come not only from the given word to be classified, but also from preceding and following words. In our experiments the same number of left and right words was taken as a context. In other words, the current word was at the centre of a window of words 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 words: the three preceding, the current, and the 3 following words.

Due to the use of a context window, the SVM input vector is the combination of the feature vector of the current word and those of the neighbouring words. The feature vector derived from a word is a long sparse vector. First, the algorithm collects all possible features from the training documents. Each feature

(e.g. a token or a part-of-speech (POS) category) corresponds to one dimension in the feature vector. So the vectors tend to have thousands of dimensions, but only with a small number of nonzero components which correspond to the features of the words. We set the value of nonzero components to 1.

As the input vector of the SVM comes from the words in a window surrounding the current word, we can weight those feature vectors from different words according to our knowledge about the relative importance of the neighbouring words. Two weighting schemes for the feature vectors from neighbouring words were investigated. The first is *equal weighting*, which keeps every nonzero component of the feature vector as 1 in the combined input vector, i.e., treats all neighbouring words as equally important. The second weighting scheme is the *reciprocal scheme*, which weights the surrounding words reciprocally to the distance to the word in the centre of the current window, reflecting the intuition that the nearer a neighbouring word is, the more important it is for classifying the given word. Formally it means that the nonzero components of the feature vector corresponding to the jth right or left neighbouring word are set to be equal to $1/j$ in the combined input vector. Therefore, we also refer to this scheme as $1/j$ weighting.

2.2 Post-processing

In our system we train two SVM classifiers for each type of entity – one classifier for the start and another one for the end word. One word entities are regarded as both start and end. In contrast, [21] trained four SVM classifiers for each named entity type – besides the two SVMs for start and end (like ours), also one for middle words, and one for single word entities. They also trained an extra SVM classifier to recognise words which do not belong to any named entity. [24] trained an SVM classifier for every possible transition of tags so, depending on the number of entities, that may lead to a large number of SVM classifiers.

As our SVM classifiers only identity the start or end word for every target class, some post-processing is needed to combine these into a single tag. We implemented a module with *three different stages* to post-precess the results from SVM classifiers:

- The *first stage* uses a simple procedure to guarantee the consistency of the recognition results. It scanned a document 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, the tags of a candidate entity are removed if the entity's length (the number of words) is not equal to the length of any entity of the same type in training set (a similar method was used in [20]).
- In contrast with the above two stages where each candidate entity is considered separately, the *third stage* puts together all possible tags for a given word and choses the best one. In detail, the output x of the SVM classifier (before thresholding) was first transferred into a probability via the Sigmoid

function $s(x) = -0.5 + 1/(1 + exp(-\beta x))$ where β was set as 2.0 in our experiments (also see [21] and [24]). Then a probability for an entity candidate was computed as $2 * s(x_s) * s(x_e)$, where x_s and x_e are the outputs of the SVM classifier for the start and end words of the candidate, respectively. Finally, for each given word, the probabilities for all possible tags were compared to each other and the tag with the highest probability P_h is assigned if P_h is greater than 0. Otherwise no tag is assigned to the word.

In our implementation the user can choose which of these three stages they want to be used during training, i.e., only the first, the first and the second, and all three. Note that both [21] and [24] used a Viterbi search algorithm as a post-procedure for their SVM classifiers, which corresponds to our third stage.

3 The Experimental Datasets

The system was evaluated on three corpora covering different IE tasks – named entity recognition (CoNLL-2003) and template filling or scenario templates [26] (seminars and jobs corpora). There were several reasons for choosing these corpora. Firstly, CoNLL-2003 provides the most recent evaluation results of many machine learning algorithms on named entity recognition. Secondly, the seminars and jobs corpora have also been used recently by many learning systems, both wrapper induction and more linguistically oriented ones (see Section 5 for a detailed discussion). Thirdly, the CONLL-2003 corpus differs from the other two corpora in two important aspects: (i) in CONLL-2003 there are many entities per document, whereas the jobs and seminar corpora have only a small number per document; (ii) CONLL-2003 documents are mostly free text, whereas the other two corpora contain semi-structured documents. Therefore, the performance of our SVM algorithm was evaluated thoroughly on these three corpora as our goal was to design a versatile approach, with state-of-the-art performance both on domain-independent IE tasks (e.g., named entity recognition) and domains-specific ones (e.g., template filling).

In more detail, the first corpus is the English part of the CoNLL-2003 shared task dataset — language-independent named entity recognition[2]. This corpus consists of 946 documents for training, 216 documents for development (e.g., tuning the parameters in learning algorithm), and 231 documents for evaluation (i.e., testing), all of which are news articles taken from the Reuters English corpus (RCV1) [22]. The corpus contains four types of named entitis — person, location, organisation and miscellaneous names.

The other two corpora are the CMU seminar announcements and the software job postings[3], in both of which domain-specific information was extracted into a number of slots. The seminar corpus contains 485 seminar announcements and four slots – start time (stime), end time (etime), speaker and location of a seminar. The job corpus includes 300 computer related job advertisements and

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

[3] Available from http://www.isi.edu/info-agents/RISE/repository.html.

17 slots such as title, salary and recruiter of the job and computer language, and application and platform required by job.

Table 1 shows the statistics for the CoNLL-2003, seminars and jobs datasets, respectively. We can see that the non-annotated words are much more than the annotated words, particularly for domain-specific datasets like seminar announcements and software job postings.

Table 1. Number of examples for each entity/slot type, together with the number of non-tagged words, in CoNLL-2003 corpus, seminars announcements, and software jobs postings, respectively.

Conll03	LOC	MISC	ORG	PER	Non-entity	
Training set	7140	3438	6321	6600	191627	
Test-a set	1837	922	1341	1842	47926	
Test-b set	1668	702	1661	1617	43654	
Seminars	Stime	Etime	Speaker	Location	Non-entity	
	980	433	754	643	157647	
Jobs	Id	Title	Company	Salary	Recruiter	State
	304	457	298	141	312	462
	City	Country	Language	Platform	Application	Area
	659	345	851	709	590	1005
	Req-years-e	Des-years-e	Req-degree	Des-degree	Post date	Non-entity
	166	43	83	21	302	127302

Machine learning systems typically separate the corpus into training and test sets. Since the CoNLL-2003 corpus already has the training, development and test set pre-specified, the system is trained on the training set, different experimental settings are tested on the development set, and the optimal ones are used to obtain the final results on the test set, which are then used for comparison with other systems.

The other two corpora do not provide such different sets, therefore training and testing need to be carried out differently. In our experiments we opted for splitting the corpora into two equal training and test sets by randomly assigning documents to one or the other[4]. In order to obtain more representative results, we carried out several runs and the final results were obtained by averaging the results from each run. Many of the learning systems evaluated on these corpora used the same approach and we adopted it to facilitate comparison (see Section 4.2).

All corpora were also pre-processed with the open-source ANNIE system, which is part of GATE [11]. This enabled us to supply our system with a number of linguistic (NLP) features, in addition to information already present in the document such as words and capitalisation information. The NLP features

[4] As the total number of documents in the seminar corpus is 485, we randomly split the dataset into 243 training documents and 242 testing ones.

are domain-independent and include token kind (word, number, punctuation), lemma, part-of-speech (POS) tag, gazetteer class, and named entity type according to ANNIE's rule-based recogniser[5]. The following section discusses the experiments using our system on the three corpora.

4 Experimental Results

This section presents the experimental results on the three datasets described above. As already discussed in Section 2, two SVM classifiers were trained for each entity and slot filler, one for the start and one for the end words. The resulting models were then run on the test set and the post-processing procedures described in Section 2 were applied. The procedure described in [23] were employed to obtain the solution of the SVM with uneven margins by solving an SVM problem. We used the SVM package SVMlight version 3.5[6] for solving the SVM. Unless stated otherwise, the default values of the parameters in SVMlight 3.5 were used.

The results below are reported using the F_1-measure, which is the harmonic mean of precision and recall. In other words, $F_1 = (2 * precision * recall)/(precision + recall)$, where *precision* is the percentage of correct entities found by the system and *recall* is the percentage of entities in the test set which are found by the system. A tag is considered correct if it matches exactly the human-annotated tag, both in terms of its type and its start and end offset in the document.[7]

The overall performance of the algorithm on a given corpus can be obtained in two different ways. One is the so called *macro-averaged* F_1, which is the mean of F_1 of all the entity types or slots in the corpus. The other is the *micro-averaged* measure[8], obtained by adding together the recognition results on all entity types first and then computing precision, recall, and F-measure. Some researchers argue that the macro-averaged measure is better than the micro-averaged one (see e.g. [30]), because the micro-averaged measure can be dominated by the larger classes so that it reflects less the performance of the algorithm on smaller classes. On the other hand, if all classes are of a comparable size, as is often the case in IE datasets, then the macro-averaged measure is not very different from the micro-averaged one. In addition, commonly used IE evaluation tools such as the MUC scorer [26] tend to use the micro-averaged measure. We use macro-averaged

[5] We also investigate the effect of different NLP features. We will present those results concerning different kinds of NLP features in another paper.

[6] Available from http://www.joachims.org/svm_light

[7] Although the results taking account both exact and partial match are informative in some applications, we only report results using exact match in order to make our results comparable to those reported on the CoNLL-2003 corpus, as well as in other previous work. In order to get an estimate of the influence of partial matching, we carried out some experiments which showed that partial match resulted in additional $0.01 - 0.03$ F_1 over the results from exact match only.

[8] See http://www.itl.nist.gov/iaui/894.02/related_projects/muc/muc_sw/muc_sw_manual.html

F-measure in Section 4.1 where we need to obtain an overall measure of the system's performance, e.g., for the purpose of establishing the impact of different parameters on the system's performance. In Section 4.2 the macro-averaged F-measure is used for comparing the overall performance of our system with the seminars and jobs datasets, while the the micro-averaged F-meauser is used on the CoNLL-2003 dataset since it was used in the CoNLL-2003 share task.

4.1 Influence of Different Parameters in the Algorithm

As part of the system evaluation, we conducted a series of experiments to investigate the influence of different algorithm settings. The first group of experiments looked into different window sizes. Then we tested different SVM kernels and values of the uneven marginal parameter. Finally we compare the two weighting schemes and the three stages of the post-processing introduced in Section 2.

In order to avoid testing each possible setting against all others, the optimal setting obtained from one group of experiments was used in subsequent experiments. However, while keeping the number of experiments down, this kind of sequential optimisation may not result in the most optimal parameter settings. For example, the optimal window size from the first group of experiments using linear kernel may not be optimal for the later experiments using quadratic kernel. Hence, the optimal setting obtained at each stage may not be the global optimal value, although we believe they are near each other.

Window size. We first did experiments using different window sizes on the three datasets. All NLP features, discussed in Section 3 were used together with word and capitalisation information. Table 2 presents the results for window size between 0 to 6. As can be seen, the results improve substantially when the window size changes from 0 to 1, which confirms that context is important in IE. The results also show that different datasets have different optimal window sizes – 5 for the seminars, 3 for the jobs, and 2 for the CoNLL-2003 dataset. Therefore, all subsequent experiments used the window size most optimal for the given dataset. In actual fact, even within the same dataset, different entities/slots seem to have different optimal window sizes. For example, although 5 is the best window size for the overall performance on the seminars corpus, the F-measure for the location slot at window size 3 (0.845) is higher than the one for window size 5 (0.816). However, in order to simplify the experimental settings, the overall optimal window sizes were used on each corpus.

Table 2. Different window sizes: macro-averaged F_1 on the three datasets. SVM with linear kernel, equal weighting, and all NLP features were used.

Dataset	0	1	2	3	4	5	6
CoNLL03 shared task	0.732	0.863	0.873	0.863	0.859	0.853	0.843
Seminar announcements	0.261	0.645	0.735	0.809	0.840	0.842	0.824
Job advertisements	0.479	0.748	0.754	0.774	0.748	0.762	0.743

SVM Model selection. The next set of experiments focused on testing different parameters of the SVM algorithm. We used the same features as above, equal weighting, and the optimal window size for each corpus, that was determined in the previous experiments. Three types of SVM kernels were compared, namely, linear, quadratic, and cubic kernels (see e.g. [21] and [24]). The results are presented in Table 3, which shows once again that different datasets have different optimal kernels. Among the three kernels, the quadratic kernel is the best for the CoNLL-2003 dataset, while the linear kernel is the best for the other two datasets.

Table 3. Results of three kernels for the SVM: macro-averaged F_1 on the three datasets.

Dataset	Linear Kernel	Quadratic kernel	Cubic kernel
CoNLL03 shared task	0.873	0.883	0.875
Seminar announcements	0.842	0.827	0.795
Job advertisements	0.774	0.737	0.699

An SVM parameter that also affects the performance is the uneven marginal parameter τ (see Section 2). As already discussed, τ is the ratio of negative margin to positive margin. The optimal value of τ is dependent on the dataset. In order to check the effect of uneven marginal parameter, we carried out experiments with two values for τ: $\tau = 0.4$ and $\tau = 1$. The former was the optimal value of τ for the document classification in [23], whle the latter reduced the learning algorithm to the original SVM. We did the experiments for the linear and quadratic kernels, respectively, as they were the ones that performed best. The results are presentedin Table 4. We can see that $\tau = 0.4$ gave better results than the original SVM with $\tau = 1$ in 3 out of 4 cases on the seminars and jobs corpora but had slightly worse result on the CoNLL-2003 dataset. Table 4 also shows that the quadratic kernel yields better results than the linear kernel for $\tau = 0.4$ on the CoNLL-2003 and seminars datasets, whereas the linear kernel is better for the jobs corpus. Therefore, in the following experiments of this subsection, the quadratic kernel with $\tau = 1$ was used on the CoNLL-2003, the quadratic kernel with $\tau = 0.4$ was used on seminars datasets, and the linear kernel with $\tau = 0.4$ was used on the jobs dataset.

Table 4. The SVM uneven marginal parameter τ. Macro-averaged F_1 results with linear and quadratic kernels on the three datasets.

Dataset	Linear kernel		Quadratic kernel	
	$\tau = 1$	$\tau = 0.4$	$\tau = 1$	$\tau = 0.4$
CoNLL03	0.873	0.870	0.883	0.877
Seminar	0.842	0.835	0.827	0.854
Job	0.774	0.789	0.737	0.781

Two weighting schemes. Table 5 presents the results for two weighting schemes – the *equal weighting* and the *reciprocal weighting* of neighbouring words. In the former, all the features of the current word as well as neighbouring words are weighted equally. In the latter, the features of neighbouring word are weighted reciprocally to the distance to the current word. Table 5 shows that the reciprocal scheme produced better results than equal weighting on the CoNLL-2003 data in two NLP feature sets, and performed better in one of the two feature sets on the other two corpora. Consequently, $1/j$ weighting scheme is used for the subsequent experiments on the three corpora.

Table 5. Two weighting schemes: macro-averaged F_1 on the three datasets. The results are on two sets of NLP features, respectively. The first feature set denoted by WCTL includes the four NLP features, word, case information, tokenkind and Lemma. The second feature set added the gazetteer and named entity recognition information to the first feature set.

Dataset	WCTL		WCTL + Gaz + NE	
	Equal weighting	$1/j$ weighting	Equal weighting	$1/j$ weighting
CoNLL03	0.863	0.873	0.883	0.890
Seminar	0.845	0.838	0.861	0.867
Job	0.789	0.804	0.796	0.787

Three stages of the post-processing procedures Table 6 presents results for the three tages of post-processing procedure discussed in Section 2. In brief, the first procedure removes the spurious start or end tags. The second procedure evaluates the lengths of candidate tags and removes a candidate if its length is not equal to the length of any tag of the same type in the training set. The third procedure considers all tags simultaneously and outputs the one with the highest probability. Table 6 presents the results of the three procedures with two NLP feature sets on the three corpora. The first feature set includes the basic NLP features: the word itself, case information, token kind and lemma. The second one includes two additional features based on the gazetteer and the named entity recognition system ANNIE.

The results in Table 6 show that the second procedure has better performance than the first on all datasets. However, the third procedure is better than the second in most cases. Therefore, subsequent experiments will use the third post-processing procedure, i.e., output the tag with the highest probability.

4.2 Comparison to Other Systems

In this subsection we compare our system with others on the three datasets. Since our system uses the NLP features produced by GATE and the learning algorithm based on SVM, we call our system **GATE-SVM**. In the experiments described in this subsection we would use the same settings as other systems in order to make a fair comparison.

Table 6. Three post-processing procedures: macro-averaged F_1 on three datasets. The results are on two sets of NLP features, respectively. Proc1, Proc2 and Proc3 denote the three post-processing procedures described in the text.

Dataset	WCTL			WCTL+Gaz+Entity		
	Proc1	Proc2	Proc3	Proc1	Proc2	Proc3
CoNLL03	0.8731	0.8757	0.8756	0.8895	0.8912	0.8918
Seminar	0.8382	0.8400	0.8400	0.8675	0.8677	0.8686
Job	0.8045	0.8054	0.8050	0.7871	0.7889	0.7891

Note that [27] presented the estimated significance boundaries for the results of many systems on the CoNLL-2003 corpus, which was computed via bootstrap sampling method. The significance boundaries can be used to determine if one result is significantly different from other results. We used the significance interval presented in [27] when we compare our results with others on the CoNLL-2003 dataset. However, unfortunately, the significance boundaries are not available for the previous results on the seminars and jobs corpora. Therefore, we estimated the significance boundaries for our results on the two datasets by also using the bootstrap sampling method (see [13]). In detail, for one experiment, 1999 random repetition samples of documents had been chosen and the distribution of F_1 in these samples was assumed to be the distribution of the performance of the experiment. As in [27], we chose the significance boundaries as the left and right boundaries of the interval with centered 90% of the distribution of the F_1 value.

Named Entity Recognition We first evaluated our system on the CoNLL-2003 dataset. Since there was a development set for tuning the learning algorithm, we tried different settings to obtain the best performance on the development set. Once again we only tested the different SVM kernel types, the window sizes, and the uneven margin parameter τ. We found that the quadratic kernel, window size 4 and $\tau = 0.5$ produced best results on the development set[9]. The $1/j$ weighting scheme and the probability post-processing procedure were used.

Table 7 presents the best results of our algorithm on the CoNLL-2003 dataset, together with the results of the top system in the CoNLL-2003 share task evaluation [15] and another participating SVM-based system [24]. Our system outperformed the other SVM-based system but is slightly worse than the best result. Compared to the summarised results in [27], our overall result is slightly better than the third best system that participated in the original CoNLL-2003 evaluation but there is no significant difference between the two results.

Template Filling The results on the seminar corpus are available for quite a few systems. Those include rule learning systems such as SRV [19], Whisk [28], Rapier [2], BWI [20], SNoW [25] and $(LP)^2$ [7], as well as statistical learning

[9] Note that the optimal values of window size and τ were different from their values obained in Section 4.1 (namely, window size as 2 and $\tau = 1$), which showed that the optimal values of learning parameters selected through the sequential optimisation might not be global optimal.

Table 7. Comparison with other systems on CoNLL-2003 shared task: F-measure on each entity type and the overall micro-averaged F-measure. The macro-averaged F-measure is also included for comparison. Test-a denotes the development set and test-b – the test set.

System	test set	LOC	MISC	ORG	PER	MA F_1	Overall
GATE-SVM	test-a	**0.9370**	**0.8613**	**0.8700**	**0.9303**	**0.8996**	**0.9083**
	test-b	**0.8925**	**0.7779**	**0.8229**	**0.9092**	**0.8506**	**0.8630**
Best one	test-a	0.9612	0.8906	0.9024	0.9660	0.9301	0.9387
	test-b	0.9115	0.8044	0.8467	0.9385	0.8753	0.8876
Another	test-a	0.9375	0.8602	0.8590	0.9391	0.8990	0.9085
SVM System	test-b	0.8877	0.7419	0.7900	0.9067	0.8316	0.8467

systems such as HMM [16] and maximum entropy (MaxEnt) [4]. See Section 5 for more details about the previous work.

One problem with carrying out comparisons on the seminar corpus is that the different system used different experimental setups. The SRV, SNoW and MaxEnt systems reported results averaged over 5 runs. In each run the dataset was randomly divided into two partitions of equal size. One partition was used for training and another for test. Furthermore, for the SRV system, a third of the training set, randomly selected, was set aside 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, in each of which the dataset was randomly split approximately into two halves, one part for training and another part for testing. BWI's and HMM results were obtained via standard cross validation.

The GATE-SVM results reported here are the average over ten runs, following the methodology of Rapier and $(LP)^2$. Table 8 presents the results of our system on seminar announcements, together with the results of the other systems. As far as it was possible, we used the same features as by the other systems to enable a more informative comparison. In particular, the results listed in Table 8, including our system, did not use any gazetteer information and named entity recogniser output. The features GATE-SVM used are words, capitalisation information, token types, lemmas, and POS tags. We just used the values of learning parameters selected in Section 4.1, (e.g. window size as 5, quadratic kernel, and $\tau = 0.4$). We computed the F_1 measure for each slot as well as the macro-averaged F_1 for overall performance for our system. Note that the majority of systems evaluated on the seminars and jobs corpora only reported per slot F-measures, without overall results. However, we think that an overall measure was useful for comparing different systems on a dataset. Hence, we computed the macro-averaged F_1 for other systems from their F_1 of every slots. The best results for each slot and the overall performance appear in bold font.

We can see that the best results on different slots were achieved by different system and the best overall performance was achieved by the $(LP)^2$. We can alse see that the results of GATE-SVM were not significantly different from the best

Table 8. Comparison with other systems on CMU seminar corpus: F_1 on each slot and overal performance. Quadratic kernel and the uneven margin parameter $\tau = 0.4$ were used in the SVM. MA F_1 refer to the macro-averaged F_1.

	Speaker	Location	Stime	Etime	MA F_1
GATE-SVM	0.690	0.813	0.948	0.927	0.845±0.030
$(LP)^2$	**0.776**	0.750	0.990	0.955	**0.868**
SNoW	0.738	0.752	**0.996**	**0.963**	0.862
MaxEnt	0.653	0.823	**0.996**	0.945	0.854
BWI	0.677	0.767	**0.996**	0.939	0.846
HMM	0.711	**0.839**	0.991	0.595	0.784
Rapier	0.531	0.734	0.959	0.946	0.791
Whisk	0.183	0.666	0.926	0.861	0.657
SRV	0.563	0.722	0.985	0.779	0.760

results for the overall performance and on most slots. If the information from the ANNIE gazetteer and named entity recogniser is used as additional features, then the micro-averaged F_1 for GATE-SVM is 0.862, which is better than the 0.857 for $(LP)^2$ using the same features (see [7]), but is still worse than the 0.872 for the maximum entropy system (see [4]). However, note that our system just used the general NLP features while [4] used genre-specific features (see Section 5 for some details). Furthermore, we did not optimise the parameter settings in the experiments specifically for this corpus (see the discussions about optimising the experimental settings for the CoNLL-2003 dataset above).

For the **jobs postings corpus**, our system was compared to two rule learning systems, Rapier and $(LP)^2$, which were evaluated on this dataset (see [2] and [7] respectively).

Again, in order to make the comparison as informative as possible, we adopted the same settings in our experiments as those used by $(LP)^2$ [8]. In particular, we executed ten runs using a random half of the corpus for training and the rest for test. The results presented here are the mean of those obtained in the ten runs. In contrast, Rapier's results were obtained via 10-fold cross validation over the entire dataset. Again, only basic NLP features are used: word, capitalisation information, token types, and lemmas. The parameter values selected for the jobs dataset in last subsection (e.g. window size as 3, linear kernel and $\tau = 0.4$) were used here. We also computed the macro-averaged F_1 for other two systems for a overall performance comparison.

Table 9 presents the results of our system as well as the results of the other two systems on the jobs corpus. GATE-SVM achieves the best results among all three on eight out of the 17 slots and the second best results on nine of the seventeen slots. Overall, the macro-averaged F_1 of GATE-SVM is better than the other systems. However, the significance boundaries indicate that the three system are not significantly different from each other.

Table 9. Comparison with other systems on the jobs corpus: F_1 on individual type of entity and the overall figure. Quadratic kernel and the uneven margin parameter $\tau = 0.4$ were used. The highest score on each slot appears in bold.

Slot	**GATE-SVM**	$(LP)^2$	Rapier	Slot	**GATE-SVM**	$(LP)^2$	Rapier
Id	0.977	**1.000**	0.975	Platform	0.801	**0.805**	0.725
Title	**0.496**	0.439	0.405	Application	0.702	**0.784**	0.693
Company	**0.772**	0.719	0.700	Area	0.468	**0.537**	0.424
Salary	**0.865**	0.628	0.674	Req-years-e	**0.808**	0.688	0.672
Recruiter	0.784	**0.806**	0.684	Des-years-e	0.819	0.604	**0.875**
State	**0.928**	0.847	0.902	Req-degree	**0.875**	0.847	0.815
City	**0.955**	0.930	0.904	Des-degree	0.592	0.651	**0.722**
Country	**0.962**	0.810	0.932	Post date	0.992	**0.995**	**0.995**
Language	0.869	**0.910**	0.818	Macro-averaged F_1	**0.808**±0.063	0.772	0.760

4.3 Mixed-Initiative Information Extraction

In mixed-initiative (or adaptive) information extraction we are concerned with the ability of an information extraction system to adapt to a new domain or application with minimum effort. From the point of the learning algorithm's view, a mixed-initiative system is required to learn an initial model from a small number of training examples. Then the performance of system would improve gradually as more and more training instances become available (e.g., from the user annotating new texts).

In order to evaluate our system in a mixed-initiative IE scenario, we evaluated the learning algorithm on a growing number of examples. For both the seminar and jobs corpora, we fist sorted documents in alphabetic order by file name. Then in each experiment the second half of corpus was used as a test set and a small number of documents were picked randomly from the first half for training. For the CoNLL-2003 dataset the training documents were chosen randomly from the training set and the results are reported on the development set. In order to factor out randomness of results, the mean of ten runs is reported. The same experimental settings were used for each of the three dataset as those in Section 4.2, respectively.

Table 10 presents the experimental results for different numbers of training documents as well as for two values of uneven marginal parameter on the three datasets, respectively. We can see that the system performance improved consistently as more training documents were used. In addition, the uneven margin parameter with value less than 1 gave better results, in particular on a small number of training documents.

Table 11 shows that some types of entities can be learned faster than others, due to their more fixed internal structure. For example, start and end times can be learned from as little as 10 documents, while at least 60 documents are required to reach similar performance on speaker and location. When interpreting these results one must bear in mind that most documents in the seminars dataset provide only one, or maximum two, examples of each slot (the ratio

Table 10. Different numbers of documents for training: macro-averaged F_1 on three datasets. The results for two different values of the uneven margin parameter are compared.

Dataset	10	20	30	40	50	60	70
	$\tau = 0.4$:						
Seminar	0.555	0.677	0.704	0.734	0.754	0.770	0.787
Job	0.434	0.509	0.539	0.575	0.597	0.608	0.607
CoNLL03	0.606	0.664	0.704	0.722	0.728	0.752	0.764
	$\tau = 1$:						
Seminar	0.377	0.485	0.572	0.621	0.633	0.683	0.701
Job	0.404	0.460	0.496	0.504	0.553	0.575	0.560
CoNLL03	0.462	0.586	0.652	0.683	0.686	0.714	0.735

Table 11. Different numbers of documents for training: macro-averaged F_1 on seminars dataset for every entity types. The uneven marginal parameter $\tau = 0.4$.

	10	20	30	40	50	60	70
stime	0.766	0.866	0.866	0.868	0.858	0.870	0.873
etime	0.783	0.839	0.873	0.882	0.881	0.875	0.877
speaker	0.275	0.448	0.540	0.558	0.584	0.628	0.633
location	0.393	0.555	0.535	0.629	0.692	0.708	0.752

between number of documents and number of examples per slot in the corpus ranges between 0.9 and 2). Therefore, in this case learning after 10 documents is almost equivalent to learning from 10 to 15 examples per slot.

Table 12. Comparison of the GATE-SVM with the $(LP)^2$: F_1 one each slot of the seminars corpus.

	Number of training docs	$(LP)^2$	GATE-SVM
stime	30	0.840	0.866
etime	20	0.823	0.839
location	30	0.700	0.535
speaker	25	0.506	0.476

Another system which carried out such experiments on the seminars dataset is $(LP)^2$ [9]. Table 12 compares our results with those of $(LP)^2$. In a nutshell, our system is better than $(LP)^2$ on etime and stime categories but is worse on location and speaker. Note that the F_1 on speaker for our system increased significantly to 0.539 if only five more training documents were added. However, we do cannot compare this to the results of $(LP)^2$ with 30 training documents, as the paper [9] does not provide this information.

5 Related Work

This section briefly describes previous work on applying machine learning to IE, in particular those systems which were evaluated on the three datasets used in our experiments. We first describe the applications of SVM to IE. Then we look at the other algorithms evaluated on the CoNLL-2003 dataset. Finally, the rule learning and statistical learning IE systems on the seminar announcements and job postings corpora are reviewed.

5.1 SVM-based Systems

The SVM based system in [21] trained four SVM classifiers for each named entity type – besides the two SVMs for start and end words like ours, one for middle words, and one for single word entities. They also trained an extra SVM classifier to recognise the words which do not belong to any named entity. [21] used a sigmoid function to transfer the SVM output into a probability and then applied the Viterbi algorithm to determine the optimal label sequence for a sentence. The system was evaluated on a Japanese IE corpus. They used the neighbouring words with window size 2. Their experiments showed that the SVM based system performed better than both maximum entropy and rule learning systems on the same dataset using the same features. They also showed that quadratic kernel was better than both linear and cubic kernels on their dataset. [21] also described an efficient implementation of the SVM with quadratic kernel.

[24] used a lattice-based approach to named entity recognition and employed SVM with cubic kernel to compute transition probabilities in a lattice. They trained an SVM classifier for every possible transition of tags, meaning that they may have a large number of SVM classifiers. They tested the system on the CoNLL-2003 dataset using cubic kernel. They also took into account the features from neighbouring words (The window size 3 was used). Their result on the CoNLL-2003 corpus is comparable to ours (see Table 7). There are some other applications of SVM for bio-named entity recognition (see e.g. [29]).

5.2 Other Learning Methods Evaluated on the CONLL'2003 corpus

CoNLL-2003 corpus is a typical named entity recognition corpus with newswire articles and entity types similar to the earlier MUC-6 and MUC-7 corpora [26]. Sixteen systems participated in the evaluation. All of them were based on statistical learning, except one system which used rule learning as one of four algorithms which were combined as one classifier. The system with the best score was exactly this combined system, based on robust risk minimisation, maximum entropy, transformation based learning and HMMs, respectively (see [15].

Another system only based on maximum entropy obtained slightly worse results (see [5]). They used quite a few features, including some genre-specific features such as the so-called zone related features which are dependent on the structure of documents. Note that another two participating systems were also only based on maximum entropy (see [1], [12]). In particular, the probability

discriminant model used in [12] was quite similar to the one in [5]. The features used in [12] were general and less than those in [5]. Both the scores of these two systems ([1] and [12]) were significantly worse than the system described in [5], which confirms the conjecture that the appropriate features are as important as the learning algorithm.

The SVM based participating system was discussed above (see [24]).

5.3 Learning Systems Evaluated on Template Filling

SRV is a relational learning (or inductive logic programming) algorithm for IE, which deduces a set of rules for one type of information entity from training examples (see [17]). It checked every text fragments of appropriate size in document in order to identify if the fragment was an information entity or not. [18, 19] tested SRV for IE on three datasets – the CMU seminars corpus, a collection of 600 newswire articles on corporate acquisitions from Reuters and a collection of web pages of university computer science departments.

WHISK [28], another relational learning system for IE, was tested on collections of structured, semi-structured and free-text documents, such as CNN weather domain, seminar announcements, software jobs postings, and news story articles. WHISK's results on the seminars corpus were not as good as SRV's, which may be attributed to the fact that WHISK used less features – only the token and its semantic class.

Rapier is also a rule based learning IE system (see [2]). It was tested on two dataset: software jobs and seminar announcements. Its results on seminar announcements are better than SRV.

BWI (Boosted Wrapper Induction) involved learning a wrapper (boundary detector) for an information entity via a boosting procedure (see [20]). It was evaluated on several collections such as seminar announcements, software job postings, Reuters articles, and web pages. [20] also considered the neighbouring words as context and found, similar to us, that different datasets have a different optimal window size. One should note that for rule based learning algorithms the training time increases exponentially with window size.

$(LP)^2$ is also a rule learning algorithm for IE (see e.g. [7]). In [9] $(LP)^2$ was tested on three datasets: seminar announcements, software job postings, and a collection of 103 web pages describing computer science courses. It compared three different sets of features. [7] also discussed the different effects of window size on different entity types.

[25] presented another relational learning based IE system, SNoW. It learned rules via a multi-class classifier by looking at a target fragment and its left and right windows. It was evaluated on the seminar announcements dataset.

[16] exploited a general statistical model, Hidden Markov Models (HMMs), for IE. It also used the shrinkage technique to deal with data spareness for HMM parameter estimations. It was tested on two corpora, the seminar announcements, and a collection of newswire articles from Reuters. It used similar experimental settings to SRV and obtained better results on the seminars corpus.

[4] used a probabilistic discriminant model for IE and used maximum entropy for parameter estimations. It was tested on several corpora including seminar announcements, the CoNLL-2003 corpus (see [5]) and the datasets from MUC-6 and MUC-7 (see [3]).

All previous work used features from a window surrounding the current word, as well as features of the word itself. [20] and [7] investigated the effect of window size on the performance of rule-based learning and noticed that the computation time increased exponentially as the window size grew. On the other hand, the computation time in an SVM based system only increases linearly with window size. Hence it is easier for the SVM algorithm to select and use the optimal window size. It also should be noted that previous systems treated the features from words in the window as equally important. In other words, this is equivalent to using the equal weighting scheme defined in Section 2. However, our experiments demonstrated that the reciprocal $1/j$ weighting achieves better results (see Section 4).

Basically the rule learning IE systems did not do any post-processing other than simple consistency checking – they treated each type of entity separately. The statistical learning algorithms compute a probability for each entity (or transfer the output into a probability as in the SVM based IE algorithms), such that they can select the best label for a fragment of text based on these probabilities. In order to select the best labels for a sentence, a Viterbi-like search algorithm was usually employed as a post-processor in the statistical learning systems.

[2] and [7] also investigated the effects of growing quantities of training data, which is useful for adaptive IE. [2] also considered active learning, where the system learns an initial model from a small pool of annotated examples and then, based on the learned model, selects additional examples for training.

6 Conclusions

This paper presents an SVM-based algorithm for IE and the experiments on three benchmark datasets, the CoNLL-2003 dataset, the CMU seminars corpus, and the software jobs corpus. The results showed that our system is comparable to other state of the art systems on both traditional IE and mixed-initiative IE tasks.

In comparison to other similar SVM-based algorithms, our algorithm is simpler, i.e., it needs a smaller number of SVM classifiers per category than the other two systems discussed respectively in [21] and [24]. GATE-SVM also obtained better results than the SVM-based system in [24] on the CoNLL-2003 corpus. In addition, our algorithm uses an uneven margin parameter, which we showed to be particularly useful for adaptive information extraction on a small amount of training data.

We investigated two weighting schemes for the features of the surrounding words and showed that the reciprocal weighting scheme performed better than the commonly used equally weighting. We also investigated three post-processing

procedures: from using the SVM outputs for begin and end tags separately to selecting the highest probability label based on the output of all SVM classifiers. We found that the probability scheme gave best results. We also carried out the experiments for investigating the influence of different algorithm settings.

References

1. Bender, O., Och, F. J., Ney, H.: Maximum entropy models for named entity recognition. In Walter Daelemans and Miles Osborne, editors, Proceedings of CoNLL-2003, 148–151. Edmonton, Canada, 2003
2. Califf, M. E.: Relational learning techniques for natural language information extraction. PhD thesis, University of Texas at Austin, 1998
3. Chieu, H. L., Ng, H. T.: A Maximum Entropy Approach to Information Extraction from Semi-Structured and Free Text. In Proceedings of the Eighteenth National Conference on Artifficial Intelligence, 786–791, 2002
4. Chieu, H. L., Ng, H. T.: Named entity recognition: A maximum entropy approach using global information. In Proceedings of the 19th International Conference on Computational Linguistics (COLING'02), Taipei, Taiwan, 2002
5. Chieu, H. L., Ng, H. T.: Named entity recognition with a maximum entropy approach. In Walter Daelemans and Miles Osborne, editors, Proceedings of CoNLL-2003, 160–163. Edmonton, Canada, 2003
6. Cimiano, P., Handschuh, S., Staab, S.: Towards the self-Annotating Web. In Proceedings of WWW'04, 2004
7. Ciravegna, F.: $(LP)^2$, an adaptive algorithm for information extraction from webrelated texts. In Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining, Seattle, 2001
8. Ciravegna, F.: $(LP)^2$, Rule Induction for Information Extraction Using Linguistic Constraints. Technical Report CS-03-07, Department of Computer Science, University of Sheffield, Sheffield, September 2003
9. Ciravegna, F., Wilks, Y.: 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
10. Cunningham, H.: Information extraction, automatic. Encyclopedia of Language and Linguistics, 2nd Edition, 2005
11. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: 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
12. Curran, J. R., Clark, S.: Language independent NER using a maximum entropy tagger. In Walter Daelemans and Miles Osborne, editors, Proceedings of CoNLL-2003, 164–167. Edmonton, Canada, 2003
13. Davison, A. C., Hinkley, D. V.: Bootstrap methods and their application. Cambridge University Press, Cambridge, UK, 1997
14. Day, D., Aberdeen, J., Hirschman, L., Kozierok, R., Robinson, P., Vilain, M.: Mixed-initiative development of language processing systems. In Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP-97), 1997
15. Florian, R., Ittycheriah, A., Jing, H., Zhang, T.: Named entity recognition through classiffier combination. In Walter Daelemans and Miles Osborne, editors, Proceedings of CoNLL-2003, pages 168–171. Edmonton, Canada, 2003

16. Freigtag, D., McCallum, A. K.: Information extraction with HMMs and shrinkage. In Proceesings of Workshop on Machine Learnig for Information Extraction, pages 31–36, 1999
17. Freitag, D.: Information extraction from html: Application of a general learning approach. Proceedings of the Fifteenth Conference on Artificial Intelligence AAAI-98, pages 517–523, 1998
18. Freitag, D.: Machine Learning for Information Extraction in Informal Domains. PhD thesis, Carnegie Mellon University, 1998.
19. Freitag, D.: Machine Learning for Information Extraction in Informal Domains. Machine Learning, **39** (2000) 169–202
20. Freitag, D., Kushmerick, N.: Boosted Wrapper Induction. In Proceedings of AAAI 2000, 2000.
21. Isozaki, H., Kazawa, H.: Efficient Support Vector Classiffiers for Named Entity Recognition. In Proceedings of the 19th International Conference on Computational Lingustics (COLING'02), pages 390–396, Taipei, Taiwan, 2002
22. Lewis, D. D., Yang, Y., Rose, T. G., Li, F.: Rcv1: A new benchmark collection for text categorization research. Journal of Machine Learning Research, **5** (2004) 361–397
23. Li, Y., Shawe-Taylor, J.: The SVM with uneven margins and Chinese document categorization. In Proceedings of The 17th Pacific Asia Conference on Language, Information and Computation (PACLIC17), pages 216–227, Singapore, Oct. 2003.
24. Mayfield, J., McNamee, P., Piatko, C.: Named entity recognition using hundreds of thousands of features. In Walter Daelemans and Miles Osborne, editors, Proceedings of CoNLL-2003, pages 184–187. Edmonton, Canada, 2003.
25. Roth, D., Yih, W. T.: 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.
26. SAIC. Proceedings of the Seventh Message Understanding Conference (MUC-7). http://www.itl.nist.gov/iaui/894.02/related projects/muc/index.html, 1998.
27. Sang, E. F., Meulder, F. D.: Introduction to the CoNLL-2003 shared task: language-independent named rntity recognition. In Walter Daelemans and Miles Osborne, editors, Proceedings of CoNLL-2003, pages 142–147. Edmonton, Canada, 2003.
28. Soderland, S.: Learning information extraction rules for semi-structured and free text. Machine Learning, **34** (1999) 233–272
29. Song, Y., Yi, E., Kim, E., Lee, G. G.: POSBIOTM-NER: a machine learning approach for bio-named entity recognition. In Workshop on a critical assessment of text mining methods in molecular biology, Granada, Spain (http://www.pdg.cnb.uam.es/BioLINK/workshop BioCreative 04/), 2004.
30. Yang, Y., Liu, X.: A re-examination of text categorization methods. In Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval, pages 42–49, 1999.