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. This paper presents an SVM-based learning system for information extraction (IE). One distinctive feature of our system is the use of a variant of the SVM, the SVM with uneven margins, which is particularly helpful for small training datasets. In addition, our approach needs fewer SVM classifiers to be trained than other recent SVM-based systems. The paper also compares our approach to several state-of-the-art systems (including rule learning and statistical learning algorithms) on three IE benchmark datasets: CoNLL-2003, CMU seminars, and the software jobs corpus. The experimental results show that our system outperforms a recent SVM-based system on CoNLL-2003, achieves the highest score on eight out of 17 categories on the jobs corpus, and is 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 introduction to IE and its applications). A lot of research has focused on named entity recognition, a basic task of IE, which classifies proper nouns and/or numerical information into classes such as persons, organizations, and dates. Effectively, most IE tasks can be regarded as the task of recognizing some information entities within text. IE can be useful in many applications, such as business intelligence, automatic annotations of web pages with semantic information, and knowledge management.

Over the past ten years, a number of machine learning techniques have been used for IE and they have achieved state-of-the-art results, comparable to manually engineered IE systems. When applying machine learning to IE, a learning algorithm usually learns a model from a set of examples, grouped in documents, which have been manually annotated by the user. Then the model can be used to extract information from new documents. The accuracy of the learned model usually increases with the number of training examples made available to the system. However, as manual annotation is a time-consuming process, it is important for an IE system to have good performance on small training sets.

Machine learning algorithms for IE can be classified broadly into two main categories: rule-based or relational learning on one hand, and statistical learning,

on the other. For each entity class, rule-based methods induce a set of rules from a training set, while statistical methods learn statistical models or classifiers. Some systems based on rule or relational learning are SRV [16], RAPIER [2], WHISK [26], BWI [18], $(LP)^2$ [7], and SNoW [23]). Some example statistical systems are HMMs [14], Maximal Entropy [4], and SVM [19] or [22].

These IE systems also differ from each other in the features that they use. Some use only basic features such as token string, capitalization, and token type (word, number, etc.), e.g. BWI. In addition, others use linguistic features such as part-of-speech, semantic information from gazetteer lists, and the outputs of other IE systems (most frequently general purpose named entity recognizers). A few systems also exploit genre-specific information such as document structure, see e.g. [4]. In general, the more features the system used, the better performance it could achieve.

One of the most successful machine learning methods for IE is Support Vector Machine (SVM), which is a general supervised machine learning algorithm. It has achieved state-of-the-art performance on many classification tasks, including named entity recognition (see e.g. [19], [22]). For instance, [19] compares three commonly used methods for named entity recognition – SVM with quadratic kernel, maximal entropy, and a rule based learning system, and shows that the SVM-based system outperforms the other two. In our view, this comparison [19] is more informative than the comparison in, e.g., the CoNLL-2003 shared task (see [25]), because the former uses both the same corpus and the same features for all three systems, while in the later different systems used the same corpus but different features[1]. As already discussed above, more features usually result in better performance and therefore, it is important to use the same or similar features on the same dataset when comparing different algorithms.

This paper describes an SVM-based learning algorithm for IE and presents detailed experimental results. In contrast to similar previous work, our SVM model (see Section 2) uses an uneven margins parameter which has been shown [21] to improve the performance for document categorization (especially for small categories). Detailed experiments investigating different SVM parameters on three benchmark datasets were carried out (see Section 4.1). The experimental datasets were chosen to enable thorough comparison between our approach and other state-of-the-art learning algorithms (see Section 4.2). The learning curve of the algorithm was also evaluated by providing a small number of initial examples and then incrementally increasing the size of the training data (see Section 4.3). Section 5 covers related work.

2 The SVM Based IE System

Due to named entities often spanning more than one word, a classifier-based IE system needs to be designed to cope with this problem. In our SVM-based

[1] The 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/).

system, called GATE-SVM, two SVM classifiers were trained for each entity type – one classifier to recognize the beginning of the entity and another one for the end. One word entities are regarded as both start and end. In contrast, [19] trained four SVM classifiers for each 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 recognize words which do not belong to any named entity. Another approach is to train an SVM classifier for every possible transition of tags [22]. In this case, at least five classifiers need to be trained for every entity type: two classifiers for the two transitions between beginning and internal words, another two for the transitions between a beginning word and a non-entity word, and one for the transition from an internal word to a non-entity word. This approach also needs extra classifiers for the transitions between two entity types. Therefore, depending on the number of entities, this approach may result in a large number of SVM classifiers. Hence, in comparison, our approach is simpler than the other two SVM-based systems, in terms of requiring the lowest the number of SVM classifiers.

The rest of this section describes the other features of our system, namely the SVM algorithm, and the pre-processing and post-processing procedures.

2.1 The SVM With Uneven Margins

The GATE-SVM system uses a variant of the SVM, the SVM with uneven margins [21], which has a better generalization performance than the original SVM on imbalanced dataset where the positive examples are much less than the negative ones. The original SVM treats positive and negative examples equally such that the margin of the SVM hyperplane to negative training examples is equal to the margin to positive training examples. However, for imbalanced training data where the positive examples are so rare that they are not representative of the genuine distribution of positive examples, a larger positive margin than the negative one would be beneficial for the generalization of the SVM classifier (see detailed discussion in [21]). Therefore, [21] introduced an uneven margins parameter into the SVM algorithm. The uneven margins parameter is the ratio of the negative margin to the positive margin. By using this parameter the uneven margins SVM is able to handle imbalanced data better than the original even margins SVM model.

The uneven margins parameter has been shown previously to facilitate document classification on unbalanced training data (see [21]). Given that IE classification tasks, particularly when learning from small data sets, often involve imbalanced data (refer to Table 3 below), we expected to gain more benefits from SVM with uneven margins over the original SVM algorithm, which is confirmed in our experimental results presented in Section 4.

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 optimization problem:

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

In these equations, τ is the uneven margins parameter which is the ratio of the negative margin to the positive margin in the classifier and is equal to 1 in the original SVM. Like other parameters of learning algorithms, the value of τ can be empirically determined by, for example, n-fold cross-validation on training set or hold-out development set. Moreover, from Table 5 in Section 4 we can see that the performance of the uneven margins SVM is not sensitive to the value of the uneven margins parameter. Therefore, a reasonable estimation of the τ can help the uneven margins SVM to achieve significantly better results than the original SVM model on imbalanced data.

The solution of the quadratic optimization problem above can be obtained by solving a related SVM problem (see [21]). In other words, it is not necessary to solve the uneven margins SVM problem directly. Instead, we can solve a corresponding standard SVM problem first by using an existing SVM implementation and then obtain the solution of uneven margins SVM through a transformation.

2.2 The Feature Vector Input to the SVM

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 is adopted in this work, which effectively means that each word is regarded as a separate instance by the SVM classifier. First of all, each document is processed using the open-source ANNIE system, which is part of GATE[2] [11]. This produces a number of linguistic (NLP) features. The features include token form, capitalization information of words, token kind, lemma, part-of-speech tag, semantic classes from gazetteer lists, and named entity type according to ANNIE's rule-based recognizer. Table 1 shows an example of text with its associated NLP features. Note that a token may not have all the NLP features considered, e.g. the token "Time" did not have the Lookup feature because it does not occur in ANNIE's gazetteer lists.

Given this input, a feature vector was derived from the NLP features of each token in the following way:

1. All possible features from the training documents are collected and indexed with a unique identifier, and each dimension of the feature vector corresponds

[2] Available from http://www.gate.ac.uk/

Table 1. NLP Features for the text sample "Time: 3:30 PM". The features are token form, capitalization information (Case), simple token kind (Tokenkind), part-of-speech (Pos), semantic classes from gazetteer lists (Lookup), and named entity types according the ANNIE. The Unknown type for word "Time" meant that ANNIE identified the word "Time" as a named entity but could not recognize its type of entity.

Token	Case	Tokenkind	Lemma	Pos	Lookup	Entity
Time	upperInitial	word	time	NNP		Unknown
:		punctuation	:	:		
3		number	3	CD		Time
:		punctuation	:	:		Time
30		number	3	CD		Time
PM	allCaps	word	pm	NNP	time	Time

to one feature (e.g. a given token string such as "Time" or a part-of-speech (POS) category such as "CD").

2. For each token, each component of the feature vector that corresponds to the value of the respective NLP feature are set to 1, and all other components are set to 0.

Table 2 presents the feature vectors for the tokens listed in Table 1. We can see that the vectors are very sparse – only several components out of an approximately 20000 dimensional vector are non-zero.

Table 2. Feature vectors for the tokens of text "Time: 3:30 PM". The vectors are presented in a compact form, i.e. only the indexes and values of all nonzero components are shown. Refer to Table 1 for the NLP features.

Token	Feature Vector
Time	4835:1 11811:1 11815:1 19009:1 19697:1 19780:1
:	399:1 11816:1 12213:1 19682:1
3	187:1 11818:1 12001:1 19685:1 19778:1
:	399:1 11816:1 12213:1 19682:1 19778:1
30	188:1 11818:1 12002:1 19685:1 19778:1
PM	3621:1 11812:1 11815:1 17292:1 19697:1 19752:1 19778:1

Since in information extraction the context of the word is usually as important as the word itself, the SVM input vector needs to take into account features of the preceding and following words, in addition to those of the given word. 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 3 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 its neighboring words.

As the input vector of the SVM combines the feature vectors of all words in the context window, these vectors can be weighted differently, depending on the relative importance of the neighboring words. In this work, two weighting schemes for the feature vectors from neighboring words were investigated. The first is the commonly used *equal weighting*, which keeps every nonzero component of the feature vector as 1 in the combined input vector, i.e., treats all neighboring 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 neighboring 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 neighboring 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**.

2.3 Post-processing the Results from the SVM Classifiers

As we train two different SVM classifiers to identity the start or end word for each target class, some post-processing is needed to combine these into a single tag. Therefore, our system has a module with *three different stages* to post-process the results from the SVM classifiers:

- The *first stage* uses a simple procedure to guarantee the consistency of the recognition results. It scans 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 the training set (a similar method was used in [18]).
- In contrast to the above two stages where each candidate entity is considered separately, the *third stage* puts together all possible tags for a given word and chooses the best one. In detail, the output x of the SVM classifier (before thresholding) is first transferred into a probability via the Sigmoid function $s(x) = 1/(1 + exp(-\beta x))$ where β is set to 2.0 (also see [19] and [22]). Then a probability for an entity candidate is computed as $s(x_s) * s(x_e)$, 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 are compared to each other and the tag with the highest probability P_h is assigned if P_h is greater than 0.25. Otherwise no tag is assigned to the word.

The three stages of the procedure can be applied sequentially to process the outputs of the classifiers. On the other hand, we can also obtain several post-processing procedures which consist of, e.g. only the first, or the first and the

second, or all three stages. We will compare the different kinds of post procedures in Section 4.1. Note that both [19] and [22] used a Viterbi search algorithm as a post-procedure for their SVM classifiers, which corresponds to applying the first and third stages of our algorithm.

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 [24] (seminars and jobs corpora). There are several reasons for choosing these corpora. Firstly, CoNLL-2003 provides the most recent detailed evaluation results of 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[3]. 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) [20]. The corpus contains four types of named entities — person, location, organization and miscellaneous names.

The other two corpora are the CMU seminar announcements and the software job postings[4], in both of which domain-specific information is 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 the seminar. The job corpus includes 300 computer related job advertisements and 17 slots encoding job details, such as title, salary, recruiter, computer language, application, and platform.

Table 3 shows the statistics for the CoNLL-2003, seminars and jobs datasets, respectively. As can be seen from that table, the non-annotated words are much more than the annotated words, particularly for domain-specific datasets like seminar announcements and software job postings. In other words, all three corpora are imbalanced datasets where the number of positive examples is much lower than the negative ones.

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

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

Table 3. 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 in the run on the test set in order to obtain the performance results, which are then used for comparison to 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[5]. 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 [11], in order to obtain the linguistic (NLP) features used in the SVM input vector, as discussed in Section 2.2 above. These features are used in addition to information already present in the documents such as words and capitalization information. The NLP features 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 recognizer[6]. The following section discusses the experimental results on the three corpora.

[5] 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.

[6] We also investigated the effect of the different NLP features, but due to space limitations the results will be included in another paper.

4 Experimental Results

As already discussed in Section 2, two SVM classifiers are trained for each entity or slot filler, one for the start and one for the end words. The resulting models are then run on the test set and the post-processing procedures described in Section 2 are applied. The algorithm described in [21] is used to obtain the solution of the SVM with uneven margins by solving an SVM problem. More specifically, the SVM package SVMlight version 3.5[7] is used for solving the SVM problem. Unless otherwise stated, the default values of the parameters in SVMlight 3.5 are 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.

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. [28]), 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. Therefore, we use macro-averaged F-measure in Section 4.1 where an overall measure of the system's performance is needed, 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 micro-averaged F-measure is used on the CoNLL-2003 dataset since it was used in the CoNLL-2003 shared task.

4.1 Influence of Different Parameters on the Algorithm's Performance

First of all, we carried out some preliminary experiments to determine the optimal parameter settings for each of three datasets. In order to avoid testing each possible setting against all others, the different settings of the parameters are investigated sequentially. The (possibly only slightly advantageous) best setting

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

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

obtained for one parameter from the current experiment was used in subsequent ones.

The first group of experiments is for different sizes of the context window, while linear kernel and all NLP features are used. Then other parameters are investigated sequentially, namely the impact of SVM kernels (linear, quadratic and cubic), two values of the uneven margin parameter τ (1.0 and 0.4), different combinations of NLP features, two weighting schemes for the features from neighboring tokens, and finally three post-processing procedures derived from the three post-processing stages discussed in Section 2. In this way, the optimal settings listed in Table 4 were obtained (note the difference for the different corpora).

Table 4. The optimal settings of system for the three datasets, obtained by the sequentially optimal experiments. For NLP features "all" means all the NLP features obtained from the ANNIE system. For post-processing "all" means that all the three stages of the procedure are used.

Setting	window size	SVM kernel	τ	NLP features	Weighting	Post-processing
Conll03	2	quadratic	0.4	all except POS	$1/j$	all
Seminars	5	quadratic	0.4	all	$1/j$	all
Jobs	3	linear	0.4	all except POS	$1/j$	all

It should be noted that while keeping the number of experiments down, such sequential optimization may not result in the most optimal parameter settings. For instance, 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 the differences to be quite small.

Next, a series of experiments was conducted to investigate the influence of the different parameters. In these experiments, we used different settings of one parameter and adopted the values of all other parameters, as presented in Table 4 for each dataset. Due to space limitations, this paper focuses on the experimental results for the unique features in our system, namely the uneven margins parameter τ, the reciprocal weighting scheme, and the post-processing procedures.

As already discussed above, on the CoNLL-2003 dataset, the system is trained on the train set and the results are reported on the development set. Each of other two datasets is split randomly in two, with one partition used for training and the other for testing, and the results are averaged over ten runs.

Uneven Margins Parameter. Our system uses the uneven margins SVM model, while other SVM-based systems for IE use the original SVM algorithm with even margins (see e.g. [19] and [22]). As discussed in Section 2, the uneven margins parameter τ is the ratio of the negative margin to the positive margin. If $\tau = 1$, the uneven margins SVM is equivalent to the original SVM model.

As already discussed, the uneven margins parameter helps the SVM handle imbalanced training sets, i.e., sets where the positive training examples are much rarer compared to the negatives ones (a common problem in classification for IE).

Table 5 presents the results for different values of uneven margins parameter for the three datasets. Firstly, the SVM with uneven margins ($\tau < 1.0$) performs statistically significantly better than the original SVM ($\tau = 1.0$) on the two datasets – Seminars and Jobs. On the other hand, the uneven margins model obtains only marginal improvements over the even margins model on the CoNLL-2003 data. This is because the classification problems on the first two corpora are much more imbalanced than those on the CoNLL-2003 dataset. The more imbalanced a classification problem is, the more helpful the uneven margins parameter is. Also see the results for small training sets in Section 4.3. Secondly, it can be seen that the results for the τ in an interval (e.g. the interval (0.4, 0.6)) are quite similar, showing that the performance is not particularly sensitive to the value of τ. Finally, $\tau = 0.6$ achieves slightly better results than $\tau = 0.4$ on the Jobs data. However, due to the small difference, all other experiments presented in this paper preserve the experimental settings from Table 4, meaning that $\tau = 0.4$ is used on both the Seminar and Jobs datasets.

Table 5. Results for different settings of uneven margins parameter of the SVM: macro-averaged F_1 (%) on the three datasets. The standard deviation is shown in parenthesis, indicating the statistical significances of the results. The best performance figures on each dataset appear in bold.

τ	1.0	0.8	0.6	0.4	0.2	0.0
Conll03	89.0	89.6	**89.7**	89.2	85.3	65.6
Seminars	81.7(±0.6)	84.0(±0.7)	85.8(±0.8)	**86.2(±0.8)**	82.6(±1.0)	55.4(±1.4)
Jobs	79.0(±1.4)	79.9(±1.2)	**81.0(±0.9)**	80.8(±1.0)	79.0(±1.3)	57.7(±1.5)

Two Weighting Schemes. We compared two weighting schemes for combining the features from surrounding words – the commonly used *equal weighting* and the *reciprocal weighting* of features from neighboring words (discussed in Section 2.2 above). Table 6 presents the results of the two weighting schemes on the three datasets. While the reciprocal scheme produces slightly better results than equal weighting on all the three datasets, the difference is not statistically significant.

Post-processing Procedures As discussed in Section 2.3, a three-stage post-processing procedure is used to combine the results of the SVM classifiers. In brief, in the first stage filters out the spurious start or end tags. The second removes entities with length not equal to the length of any example tag in the training set. The third stage outputs the category with the highest probability. Based on these three different filtering strategies, three post-processing procedures were experimented with: the first strategy only; the first and second

Table 6. Two weighting schemes: macro-averaged F_1 (%) on the three datasets. In bold are the best performance figures for every dataset.

	Equal weighting	$1/j$ weighting
Conll03	88.4	**89.2**
Seminars	85.5(±1.0)	**86.2(±0.8)**
Jobs	80.5(±1.0)	**80.8(±1.0)**

strategies; and finally all three in sequence. In the first and second procedures, if one piece of text is assigned more than one tag, then the last tag according to the tag order in Table 3 is assigned.

Table 7 presents the results before post-processing as well as the results for the three procedures on the three corpora. Compared to the results with no post-processing, the results are improved significantly by post-processing. However, procedures 2 and 3 only obtain slightly better results than their previous stages.

Table 7. Comparison of the results without and with the post-processing procedures: macro-averaged F_1 (%) on the three datasets. Proc1, Proc2 and Proc3 denote respectively the three post-processing stages. In bold are the best performance figures for every dataset.

	No post-processing	Proc1	Proc2	Proc3
Conll03	87.5	89.0	89.1	**89.2**
Seminars	81.7(±1.0)	85.5(±0.9)	85.7(±0.9)	**86.2(±0.8)**
Jobs	77.0(±0.7)	79.9(±0.9)	80.3(±0.9)	**80.8(±1.0)**

4.2 Comparison to Other Systems

This section compares our system to other machine learning approaches 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 used similar settings to those in the other systems, in order to enable a fair comparison.

The significance boundaries of the results on the CoNLL-2003 corpus, presented in this paper, are estimated via bootstrap sampling method [25] and can be used to determine if a result is significantly different than all others.

For the other two datasets, the significance boundaries of previous results are not available. Since we ran ten experiments on each of the two datasets, the standard deviations from the results of the ten experiments are used as the significance measure in the comparison to other systems.

Named Entity Recognition The performance of GATE-SVM on named entity recognition is evaluated on the CoNLL-2003 dataset. Since this set comes with development data for tuning the learning algorithm, different settings were

tried in order to obtain the best performance on the development set. The different SVM kernel types, window sizes, and values of the uneven margin parameter τ were tested. The results showed that quadratic kernel, window size 4 and $\tau = 0.5$ produce best results on the development set. These optimal values are slightly different from those obtained in Section 4.1, which shows that the values for learning parameters selected through sequential optimization may not be globally optimal. The $1/j$ weighting scheme and all three post-processing stages are used.

Table 8. Comparison to other systems on the CoNLL-2003 shared task: F-measure (%) on each entity type and the overall micro-averaged F-measures. The macro-averaged F-measure (MA F_1) is also included for comparison. Test-a denotes the development set and test-b is as the test set. The only difference between GATE-SVM-1 and GATE-SVM-2 is in the NLP feature used (see text). The best performance figures for each entity type and overall appear in bold.

System	test set	LOC	MISC	ORG	PER	MA F_1	Overall
GATE-SVM-1	test-a	93.70	86.13	87.00	93.03	89.96	90.83
	test-b	89.25	77.79	82.29	90.92	85.06	86.30
GATE-SVM-2	test-a	93.46	86.36	86.76	92.24	89.70	90.49
	test-b	89.20	77.66	81.60	90.68	84.78	86.00
Best one	test-a	**96.12**	**89.06**	**90.24**	**96.60**	**93.01**	**93.87**
	test-b	**91.15**	**80.44**	**84.67**	**93.85**	**87.53**	**88.76±0.7**
Participating	test-a	93.75	86.02	85.90	93.91	89.90	90.85
SVM Based System	test-b	88.77	74.19	79.00	90.67	83.16	84.67±1.0

Table 8 presents the results of our system on the CoNLL-2003 dataset, together with the results of the top system in the CoNLL-2003 share task evaluation [13] and another participating SVM-based system [22], which are taken from the summary paper [25]. The results of our systems are given using two different settings in order to make a fairer comparison to the SVM based system entered in the shared task. GATE-SVM-1 uses all NLP features obtained from ANNIE, except part-of-speech information. The only difference between GATE-SVM-1 and GATE-SVM-2 is that GATE-SVM-2 does not use the semantic information from ANNIE's gazetteer lists. Therefore, all types of NLP features used by GATE-SVM-2 were also used by the participating SVM-based system. The results of both GATE-SVM-1 and GATE-SVM-2 are significantly better the participating SVM-based system. However, our results are significantly worse than the best result, which was obtained by combining the outputs of four different classifiers and other information.

Template Filling The seminar corpus has been used to evaluate quite a few learning systems. Those include rule learning approaches such as SRV [17], Whisk [26], Rapier [2], BWI [18], SNoW [23] and $(LP)^2$ [7], as well as statistical learning systems such as HMM [14] and maximum entropy (MaxEnt) [4]. See Section 5 for more details.

The major problem with carrying out comparisons on the seminar corpus is that the different systems used different experimental setups. For instance, SRV, SNoW and MaxEnt reported results averaged over 5 runs. In each run the dataset was randomly divided into two partitions of equal size – one used for training and one for testing. Furthermore, SRV used a randomly selected third of the training set for validation. WHISK's results were from 10-fold cross validation on a randomly selected set of 100 documents. Rapier's and $(LP)^2$'s results were averaged over 10 runs, instead of the 5 runs used in SRV, SNoW and MaxEnt. Finally, 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 9 presents the results of our system on seminar announcements, together with the results of the other systems. As far as it was possible, we use the same features as the other systems to enable a more informative comparison. In particular, the results listed in Table 9, including our system, did not use any gazetteer information and named entity recognizer output. The only features in this case are words, capitalization information, token types, lemmas, and POS tags. The settings for the SVM parameters were taken from Table 4, i.e., window size 5, quadratic kernel, and $\tau = 0.4$.

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

Table 9. Comparison to other systems on CMU seminar corpus: F_1 (%) on each slot and overall performance (macro-averaged F_1). Standard deviation for the MA F_1 of our system is presented in parenthesis. The best results for each slot and the overall performance appear in bold font.

	Speaker	Location	Stime	Etime	MA F_1
GATE-SVM	69.0	81.3	94.8	92.7	84.5(±0.8)
$(LP)^2$	**77.6**	75.0	99.0	95.5	**86.8**
SNoW	73.8	75.2	**99.6**	**96.3**	86.2
MaxEnt	65.3	82.3	**99.6**	94.5	85.4
BWI	67.7	76.7	**99.6**	93.9	84.6
HMM	71.1	**83.9**	99.1	59.5	78.4
Rapier	53.1	73.4	95.9	94.6	79.1
Whisk	18.3	66.6	92.6	86.1	65.7
SRV	56.3	72.2	98.5	77.9	76.0

Table 9 shows that the best results on the different slots are achieved by different system and that the best overall performance is achieved by the $(LP)^2$.

The GATE-SVM did not perform as well as the best results on the Seminar data. But it still outperformed many other systems.

However, if information from the ANNIE gazetteer and named entity recognizer is used as additional features, then the macro-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]). While this is still slightly worse than the 0.872 of the maximum entropy system (see [4]), a direct comparison between the GATE-SVM and that system cannot be made. This is due to our system just using general NLP features while [4] used genre-specific features (see Section 5 for further details). Furthermore, GATE-SVM's parameter settings were not optimized specifically for this corpus (see the discussions about optimizing the experimental settings for the CoNLL-2003 dataset above).

On the **jobs postings corpus**, GATE-SVM is compared to two rule learning systems, Rapier [2] and $(LP)^2$ [7], which are among the few evaluated on this dataset.

Again, in order to make the comparison as informative as possible, we adopted the same settings in our experiments as those used by the system which reported the highest results on this dataset, i.e., $(LP)^2$ [8]. 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. In contrast, Rapier's results were obtained via 10-fold cross validation over the entire dataset, thus making it impossible to adopt a unified approach. As in the previous experiment, only basic NLP features are used: word, capitalization information, token types, and lemmas. The parameter values of window size 3, linear kernel and $\tau = 0.4$) are used here. The macro-averaged F_1 for the other two systems is computed for overall performance comparison.

Table 10. Comparison to other systems on the jobs corpus: F_1 (%) on each entity type and overall performance as macro-averaged F_1. Standard deviation for the MA F_1 of our system is presented in parenthesis. The highest score on each slot and overall performance appears in bold.

Slot	**GATE-SVM**	$(LP)^2$	Rapier	Slot	**GATE-SVM**	$(LP)^2$	Rapier
Id	97.7	**100**	97.5	Platform	80.1	**80.5**	72.5
Title	**49.6**	43.9	40.5	Application	70.2	**78.4**	69.3
Company	**77.2**	71.9	70.0	Area	46.8	**53.7**	42.4
Salary	**86.5**	62.8	67.4	Req-years-e	**80.8**	68.8	67.2
Recruiter	78.4	**80.6**	68.4	Des-years-e	81.9	60.4	**87.5**
State	**92.8**	84.7	90.2	Req-degree	**87.5**	84.7	81.5
City	**95.5**	93.0	90.4	Des-degree	59.2	65.1	**72.2**
Country	**96.2**	81.0	93.2	Post date	99.2	**99.5**	**99.5**
Language	86.9	**91.0**	81.8	MA F_1	**80.8(±1.0)**	77.2	76.0

Table 10 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 the remaining nine. Overall, the macro-averaged F_1 of GATE-SVM is significantly better than the other two systems.

4.3 Information Extraction from Small Training Sets

The application of SVM (or other supervised learning algorithms) to IE requires a manually annotated training set. Since manual annotation is a time-consuming process. learning from small data sets is highly desirable.

Consequently, we evaluated the learning algorithm on a growing number of examples. For both the seminar and jobs corpora, a small number of documents from the corpus were selected randomly as the training set and the remaining ones were used for testing. 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 features and system parameters were used on each of the three datasets as those in Section 4.2.

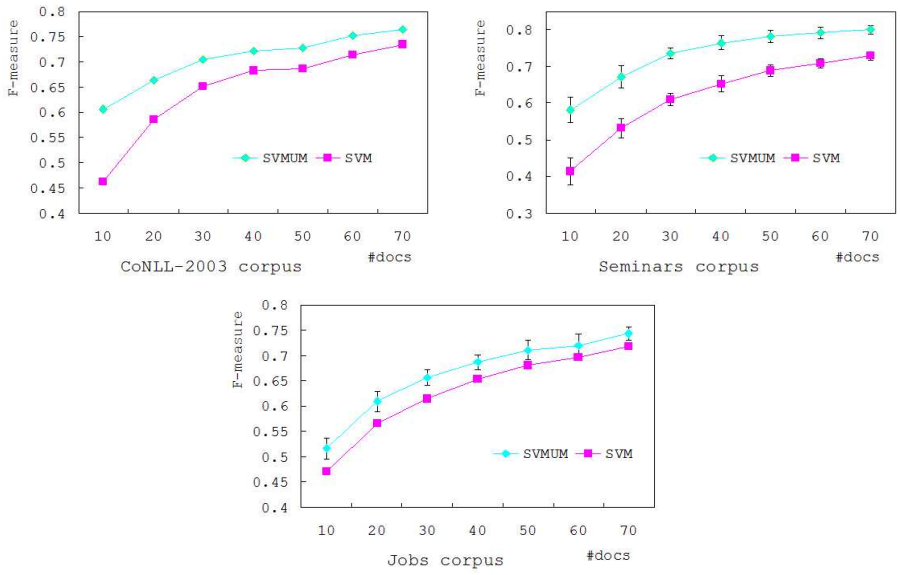

Fig. 1. Learning curves for overall F1 with a growing number of training documents on the three datasets. On each dataset the uneven margins SVM is compared to the original SVM model. For the Seminar and Jobs datasets, the error bar at a data point show the 95% confidence interval derived from the standard deviations. For clarity, we just show the confidence intervals for one curve in the Jobs graph – the confidence intervals are similar for another curve.

Figure 1 shows the learning curves of the SVM models with and without uneven margins on the three datasets. The 95% confidence intervals for the data points are also shown for the Seminars and Jobs datasets. System performance improves consistently as more training documents become available. In addition, the uneven margins SVM model demonstrates clearly better results than the original SVM, in particular on a small number of training documents.

Table 11. Different numbers of documents for training: macro-averaged F_1 (%) on seminars dataset for every entity type.

	10	20	30	40	50	60	70
stime	82.4	86.6	88.9	90.5	91.4	92.0	92.4
etime	70.7	80.6	85.9	88.1	88.5	89.2	90.6
speaker	30.7	42.4	55.6	60.6	63.4	65.5	65.9
location	48.4	58.6	63.7	67.0	69.6	69.9	70.9

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 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 GATE-SVM with $(LP)^2$: F_1 (%) for each slot of the seminars corpus and the macro averaged F_1 for overall performance. The highest score on each slot and overall performance appears in bold.

	Number of training docs	$(LP)^2$	GATE-SVM
stime	30	84.0	**88.9**
etime	20	**82.3**	80.6
speaker	25	**50.6**	50.5
location	30	**70.0**	63.7

Another system which carried out such experiments on the seminars dataset is $(LP)^2$ [9]. Table 12 compares our system with the system based on $(LP)^2$. In a nutshell, our system is better than $(LP)^2$ on the stime category but is worse on etime, speaker and location.

5 Related Work

This section briefly discusses 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 [19] 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 recognize the words which do not belong to any named entity. [19] 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 neighboring 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. [19] also described an efficient implementation of the SVM with quadratic kernel.

[22] 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 neighboring words with window size 3. Their result on the CoNLL-2003 corpus is comparable to ours (see Table 8). There are some other applications of SVM for bio-named entity recognition (see e.g. [27]).

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 [24]. 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 into one classifier. The system with the best score was exactly this combined system, based on robust risk minimization, maximum entropy, transformation based learning and HMMs, respectively (see [13].

Another system only based on maximum entropy obtained slightly worse results (see [5]). They used many different features, including some genre-specific one, 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

discriminate 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 appropriate features are at least as important as the learning algorithm.

The SVM based participating system was discussed above (see [22]).

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 [15]). It checked every text fragments of appropriate size in document in order to identify if the fragment belongs to an entity or not. [16, 17] 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 [26], 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 [18]). It was evaluated on several collections such as seminar announcements, software job postings, Reuters articles, and web pages. [18] also considered the neighboring 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]). $(LP)^2$ was tested on three datasets: seminar announcements, software job postings, and a collection of 103 web pages describing computer science courses (see [9]). It also compared three different sets of features. The effect of window size on the different entity types was also studied [9].

[23] 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.

[14] exploited a general statistical model, Hidden Markov Models (HMMs), for IE. It also used the shrinkage technique to deal with data sparseness 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 discriminate 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. Both [18] and [7] investigated the effect of window size on the performance of rule-based learning and noticed that the computation times of both the rule learning algorithms BWI and $(LP)^2$ 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. Therefore, it is easier for the SVM algorithm to select and use the optimal window size.

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. [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 show that our system is comparable to other state of the art systems for IE.

While other SVM-based IE systems used the original SVM model which treats the negative and positive margins equally, our system uses the SVM with uneven margins. Our experiments show that the uneven margins SVM performs significantly better than the original SVM on the three datasets, particularly for small training sets.

In comparison to other similar SVM-based algorithms, our system is simpler, i.e., it needs a smaller number of SVM classifiers per entity type than the other two systems discussed respectively in [19] and [22]. Our system also obtained better results than the SVM-based system in [22] on the CoNLL-2003 corpus.

We investigated two weighting schemes for the features of the surrounding words and showed that the reciprocal weighting scheme performs slightly better than the commonly used equal weighting. We also investigated several post-processing procedures, ranging 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, while overall post-processing can improve the results significantly, some of its stages only obtain small improvements.

Acknowledgements: We would like to thank the two anonymous reviewers for detailed comments and valuable suggestions. This research is supported by the EU-funded SEKT project (http://www.sekt-project.com).

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 Artificial 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 web related 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. Florian, R., Ittycheriah, A., Jing, H., Zhang, T.: Named entity recognition through classifier combination. In Walter Daelemans and Miles Osborne, editors, Proceedings of CoNLL-2003, pages 168–171. Edmonton, Canada, 2003
14. Freigtag, D., McCallum, A. K.: Information extraction with HMMs and shrinkage. In Proceedings of Workshop on Machine Learning for Information Extraction, pages 31–36, 1999

15. 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
16. Freitag, D.: Machine Learning for Information Extraction in Informal Domains. PhD thesis, Carnegie Mellon University, 1998.
17. Freitag, D.: Machine Learning for Information Extraction in Informal Domains. Machine Learning, **39** (2000) 169–202
18. Freitag, D., Kushmerick, N.: Boosted Wrapper Induction. In Proceedings of AAAI 2000, 2000.
19. Isozaki, H., Kazawa, H.: Efficient Support Vector Classifiers for Named Entity Recognition. In Proceedings of the 19th International Conference on Computational Linguistics (COLING'02), pages 390–396, Taipei, Taiwan, 2002
20. 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
21. 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.
22. 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.
23. 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.
24. SAIC. Proceedings of the Seventh Message Understanding Conference (MUC-7). http://www.itl.nist.gov/iaui/894.02/related projects/muc/index.html, 1998.
25. Sang, E. F., Meulder, F. D.: Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, Proceedings of CoNLL-2003, pages 142–147. Edmonton, Canada, 2003.
26. Soderland, S.: Learning information extraction rules for semi-structured and free text. Machine Learning, **34** (1999) 233–272
27. 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.
28. 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.