Instance Filtering for Entity Recognition

Alfio Massimiliano Gliozzo
ITC-irst
Via Sommarive, 18
38050 Trento, Italy
gliozzo@itc.it

Claudio Giuliano
ITC-irst
Via Sommarive, 18
38050 Trento, Italy
giuliano@itc.it

Raffaella Rinaldi
ITC-irst
Via Sommarive, 18
38050 Trento, Italy
rinaldiraf@itc.it

ABSTRACT

In this paper we propose *Instance Filtering* as preprocessing step for supervised classification-based learning systems for entity recognition. The goal of Instance Filtering is to reduce both the skewed class distribution and the data set size by eliminating negative instances, while preserving positive ones as much as possible. This process is performed on both the training and test set, with the effect of reducing the learning and classification time, while maintaining or improving the prediction accuracy. We performed a comparative study on a class of Instance Filtering techniques, called *Stop Word Filters*, that simply remove all the tokens belonging to a list of stop words. We evaluated our approach on three different entity recognition tasks (i.e. Named Entity, Bio-Entity and Temporal Expression Recognition) in English and Dutch, showing that both the skewness and the data set size are drastically reduced. Consequently, we reported an impressive reduction of the computation time required for training and classification, while maintaining (and sometimes improving) the prediction accuracy.

1. INTRODUCTION

The objective of Information Extraction (IE) is to identify a set of relevant domain-specific classes of entities and their relations in textual documents. In this paper we focus on the problem of Entity Recognition (ER). Recent evaluation campaigns on ER, such as the CoNLL-2002, CoNLL-2003, TERN 2004 and JNLPBA 2004 shared tasks, have shown that most of the participating systems approach the task as a supervised classification problem, assigning an appropriate classification label for each token in the input documents. However, two problems are usually related to this approach: the *skewed class distribution* and the *data set size*. The unbalanced distribution of examples, due to the entity sparseness in texts, yields in many systems a drop off in classification accuracy. In additon, supervised learning from large data sets is a time-consuming process, as the complexity of many supervised algorithms is proportional to the number of examples.

To address these problems, we propose a technique called *Instance Filtering* (IF). The goal of IF is to reduce both the skewness and the data set size by removing negative instances and preserving the positive ones. The main peculiarity of this technique is that it is performed on both the training and test sets. This reduces the computation time and the memory requirements for learning and classification, and improves the classification performance.

We present a comparative study on *Stop Word Filters*, a class of IF techniques that remove from the data set all instances (i.e. tokens) belonging to a list of stop words. The basic assumption behind this approach is as follows: stop words do not provide any specific information about the text in which they appear, therefore their expectation of being (part of) relevant entities is very low. Lists of stop words can be easily acquired by computing statistics on the training corpus. Furthermore, Stop Word Filters are easy to implement and can be successfully used as preprocessing modules by most of the supervised systems for ER, allowing to scale the IE process from toy problems to real-world applications. In this work, we compare three different metrics to select stop words: Information Content, Correlation Coefficient, and Odds Ratio.

To evaluate our filtering techniques we used the SIE system, a supervised system for ER developed at ITC-irst. SIE is designed to achieve the goal of being easily and quickly portable across tasks and languages. SIE is based on Support Vector Machines and uses a (standard) general purpose feature set.

We performed experiments on three different ER tasks (i.e. Named Entity, Bio-Entity and Temporal Expression Recognition) in two languages (i.e. English and Dutch). The results show that both the skewness and the data set size are drastically reduced, while the overall performance is increased in the bio-entity recognition task and maintained in the other tasks. Consequently, we reported a considerable reduction of the computation time required for training and classification. Overall, the system performs comparably to the state-of-the-art, demonstrating the general applicability of this technique.

The paper is structured as follows. In the next section, we discuss the background of the proposed approach and provide an overview of the related work. In Section 3 we define a general framework for IF and we introduce the class of Stop Word Filters. In Section 4 we briefly describe the SIE system. In Section 5 results of a comparison of three Stop Word Filters on several benchmark data sets are presented. Finally, Section 6 is reserved for conclusions and topics for future research.

2. BACKGROUND AND RELATED WORK

Learning with skewed class distributions is a very well-known problem in machine learning. It has been experimentally shown that an unbalanced class distribution leads to poor performance on the minority class [16] (i.e. the positive class). The error rate of a classifier trained on a skewed data set is typically very low for the majority class (i.e. the negative class), while it is unacceptable for the minority class in most of the cases. This phenomenon causes biased estimation [11] and suboptimal classification performance [2]. The poor performance on the minority class reflects on a drop off in both precision and recall [12] because both measures estimate the ability of the classifier to predict the positive class.

The most common technique for dealing with skewed data sets is sampling. It consists in reducing the skewness by altering the distribution of training examples [17]. The basic sampling methods are under-sampling and over-sampling. Under-sampling eliminates majority class examples, while over-sampling increases the minority class. IF is a technique for performing uniform under-sampling on both the training and the test sets.

An additional problem is the huge size of the data sets. In fact, to collect a sufficient amount of positive examples, it is inevitable to accumulate a large number of negative ones. The complexity of many supervised algorithms is strongly related to the data set size. For example, in the Support Vector Machines algorithm the number of support vectors increases linearly with the number of training examples [15], causing a loss of efficiency during both learning and prediction [4]. The large size of the training set is a relevant problem also for instance-based algorithms. For example, the complexity of kNN in classification is linear in the number of training examples. The problem of reducing the number of instances used for training while maintaining (and sometimes improving) the generalization accuracy, has been called Instance Pruning [18]. Instance Pruning techniques have been mainly applied to instance-based learning algorithms (e.g. kNN), to speed up the classification process while minimizing the memory requirements. The main drawback of many Instance Pruning techniques is their time complexity, which is generally quadratic in the data set size. In [19] the authors give a good overview of these methodologies. IF is a technique for performing uniform Instance Pruning on both the training and the test sets.

IF techniques have been proposed in the IE literature to alleviate the computational load and to reduce the data skewness in ER tasks. They can be considered as a way to perform instance pruning by under-sampling the negative class. In contrast to sampling and pruning, IF is performed on both the training and the test sets. Below, we will briefly review a set of IF techniques[1]. In [13] the authors approach IE as a two-step classification strategy of text fragments. The aim of the first step is to filter out most of the negative examples without eliminating positive examples, so it is an IF in its proper sense. This filter can be perceived as a classifier that achieves high recall, while the second classifier tends to high precision. The filter was able to discard around 90% of the data set instances while losing only about 2% of positive examples in a standard IE benchmark. A similar approach is reported in [5]. The authors implemented a two-step supervised algorithm for ER. The first step is performed by a "sentence filter", that discards all the sentences that are not likely to contain relevant information. Entity boundaries are then assigned to all the instances in the remaining sentences by adopting a standard supervised approach for ER. The filter is implemented by a supervised text classification system, tuned in order to maximize recall. In [14] IF is performed as a preprocessing step, by eliminating all the tokens that are not located inside a noun phrase and whose part-of-speech (PoS) belongs to a stop list. This approach was applied to the *JNLPBA* shared task, allowing about 50% of the tokens to be filtered out, while losing 0,05% of positive instances. The main limitation of this technique is its portability across languages and domains. In [8], we investigated an IF technique for IE that filters uninformative words from both the training and the test sets. Uninformative words have been identified by estimating the ratio between their probability of being (part of) an entity and their probability of lying outside. In this paper we extend and generalize such previous work.

[1]Note that the term *Instance Filtering* is introduced for the first time in this paper, while in most of the cited works filtering techniques have been referred to by different names.

3. INSTANCE FILTERING

IF is a preprocessing step performed to reduce the number of instances given as input to a supervised classifier for ER. In this section, we describe a formal framework for IF and introduce two metrics to evaluate an Instance Filter. In addition, we define the class of Stop Word Filters and propose an algorithm for their optimization.

3.1 A general framework

Let $T = (t_1, t_2, \ldots, t_n)$ be the list of all the tokens contained in a corpus T, let $\tau(t)$ be a function that returns the type w associated to the token t[2], let $V = \{w|\tau(t_i) = w\}$ be the vocabulary of T, and let $\delta(t_i)$ be a function such that

$$\delta(t_i) = \begin{cases} 1, & \text{if } t_i \text{ is a positive example;} \\ 0, & \text{if } t_i \text{ is a negative example.} \end{cases} \quad (1)$$

An Instance Filter is a function $\Delta(t_i, T)$ that returns 0 if the token t_i is not expected to be part of a relevant entity, 1 otherwise. When Δ is applied to the whole data set T, it returns a *Filtered data set* T' such that $T' = \Delta(T) = \{t_i|\Delta(t_i, T) = 1\}$. If labeled data is required to define the Filtering Function Δ, we say that Δ is a *Supervised* Instance Filter. Otherwise, when unlabeled data is used or the filter is rule-based, Δ is an *Unsupervised* Instance Filter[3]. An Instance Filter Δ can be evaluated using the two following functions:

$$\psi(\Delta, T) = \frac{|\Delta(T)|}{|T|} \quad (2)$$

[2]The terms "token" and "type" are used herein in accordance with their usual meanings in linguistics. A token is any word found in the corpus, a type is a set of identical tokens. A token is then defined by a type, together with a location at which that type occurs.

[3]Following this distinction, [14] implements an Unsupervised Instance Filter, while both [5] and [13] implement a Supervised Instance Filter.

and

$$\psi^+(\Delta, T) = \frac{\sum_{i \in \Delta(T)} \delta(t_i)}{\sum_{i \in T} \delta(t_i)} \quad (3)$$

where $\psi(\Delta, T)$ is called the *Filtering Rate* and denotes the total percentage of filtered tokens in the data set T, and $\psi^+(\Delta, T)$ is named as *Positive Filtering Rate* and denotes the percentage of positive tokens (wrongly) removed. Δ is a good filter if $\psi^+(\Delta, T)$ is minimized and $\psi(\Delta, T)$ is maximized, allowing to reduce as much as possible the data set size while preserving most of the positive instances.
In order to avoid over-fitting, the Filtering Rates among the training and test set (T_L and T_T, respectively) have to be preserved:

$$\frac{\psi^+(\Delta, T^T)}{\psi^+(\Delta, T^L)} \simeq 1 \text{ and } \frac{\psi(\Delta, T^T)}{\psi(\Delta, T^L)} \simeq 1. \quad (4)$$

In addition, we use the *skewness ratio* given by Equation 5 to evaluate the ability of an Instance Filter to reduce the data skewness.

$$\omega(T) = \frac{|T| - \sum_{t_i \in T} \delta(t_i)}{\sum_{t_i \in T} \delta(t_i)}. \quad (5)$$

We are therefore interested in Instance Filters that produce a filtered data set T' whose skewness ratio $\omega(T')$ is considerably lower than its original one:

$$\omega(\Delta(T)) \ll \omega(T). \quad (6)$$

3.2 Stop Word Filters

In this subsection, we present a class of Instance Filters called *Stop Word Filters*. They are implemented in two steps: first, Stop Words are identified from the training corpus T and collected in the set of types $U \subset V$; then all their tokens are removed from both the training and the test set[4]. Formally, a Stop Word Filter is defined by

$$\Delta_U(t_i, T) = \begin{cases} 1 & \text{if } \tau(t_i) \in U, \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

A Stop Word Filter Δ_U is fully specified by a list of stop words U. To find such a list, we borrowed from the text classification community a number of feature selection methods. In text classification, feature selection is used to remove non-informative terms from *bag-of-words* representations of texts. In this sense, IF is closely related to feature selection: in the former non-informative words are removed from the *instance set*, while in the latter they are removed from the *feature set*. Below, we present a set of metrics used to collect a stop word list U from a training corpus T. These metrics are functions of the following probabilities.

1. $P(w) = \frac{|\{t_i|\tau(t_i)=w\}|}{|T|}$,
2. $P(\bar{w}) = \frac{|\{t_i|\tau(t_i)\neq w\}|}{|T|}$,
3. $P(+) = \frac{|\{t_i|\delta(t_i)=1\}|}{|T|}$,
4. $P(-) = \frac{|\{t_i|\delta(t_i)=0\}|}{|T|}$,
5. $P(w^+) = \frac{|\{t_i|\tau(t_i)=w \wedge \delta(t_i)=1\}|}{|\{t_i|\tau(t_i)=w\}|}$,
6. $P(w^-) = \frac{|\{t_i|\tau(t_i)=w \wedge \delta(t_i)=0\}|}{|\{t_i|\tau(t_i)=w\}|}$,
7. $P(\bar{w}^+) = \frac{|\{t_i|\tau(t_i)\neq w \wedge \delta(t_i)=1\}|}{|\{t_i|\tau(t_i)\neq w\}|}$,
8. $P(\bar{w}^-) = \frac{|\{t_i|\tau(t_i)\neq w \wedge \delta(t_i)=0\}|}{|\{t_i|\tau(t_i)\neq w\}|}$.

[4]Note that only *instances* are removed from the data set, while words in U still appear in the feature description of the remaining tokens.

3.2.1 Information Content (IC)

The most commonly used feature selection metric in text classification is based on document frequency (i.e the number of documents in which a term occurs). The basic assumption is that occurrences of very frequent types in texts are non-informative for document indexing. Our approach consists in removing all the tokens whose type has a very low information content. The IC filter[5] is specified by:

$$U_\theta = \{w | P(w) \log P(w) > \theta\}. \quad (8)$$

3.2.2 Correlation Coefficient (CC)

In text classification the χ^2 statistic is used to measure the lack of independence between a type w and a category [20]. In our approach we use the correlation coefficient $CC^2 = \chi^2$ of a term w with the negative class, to find those types that are less likely to express relevant information in texts. The CC filter is specified by:

$$U_\theta = \{w | CC(w, -) > \theta\}, \quad (9)$$

where

$$CC(w, -) = \frac{\sqrt{|T|}[P(w^+)P(\bar{w}^-) - P(w^-)P(\bar{w}^+)]}{\sqrt{P(w)P(\bar{w})P(+)P(-)}}. \quad (10)$$

3.2.3 Odds Ratio (OR)

Odds ratio measures the ratio between the probability of a type to occur in the positive class, and its probability to occur in the negative class. In text classification the idea is that the distribution of the features on the relevant documents is different from the distribution on non-relevant documents [21]. Following this assumption, our approach is that a type is non-informative when its probability of being a negative example is sensibly higher than its probability of being a positive example [8]. The OR filter is specified by:

$$U_\theta = \left\{ w \middle| ln \frac{P(w^+)}{P(w^-)} > \theta \right\}. \quad (11)$$

3.3 Optimization Issues

In this section we describe how to find the optimal threshold θ for a Stop Word Filter. We derive the optimal threshold value by resolving the following optimization problem:

$$\begin{aligned} \theta_{opt} = \arg\max_\theta \quad & \psi(\Delta_{U_\theta}, T), \\ \text{subject to} \quad & \psi^+(\Delta_{U_\theta}, T) < \epsilon. \end{aligned} \quad (12)$$

To solve this problem, we observe the behaviors of ψ and ψ^+ which, for construction, are decreasing functions of θ, as shown in Figure 1. As a consequence, the optimization problem can be reformulated as the problem of finding the minimum $\theta \in \mathcal{R}$ such that $\psi^+(\Delta_{U_\theta}, T) < \epsilon$, that can be easily solved by progressively decreasing θ until the upper

[5]IC is an unsupervised filter.

bound ϵ for the positive filtering rate ψ^+ is reached[6]. Filtering Rates can be estimated by performing n-fold cross-validation on the training set.

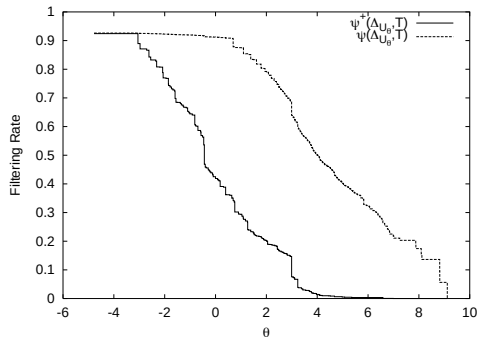

Figure 1: ψ and ψ^+ varying the threshold θ

4. A SIMPLE INFORMATION EXTRACTION SYSTEM

SIE (Simple Information Extraction) is a supervised system for Entity Recognition [7]. It casts the IE task as a classification problem by applying Support Vector Machines for detecting entity boundaries in texts. SIE is designed with the goal of being easily and quickly portable across different tasks and languages.

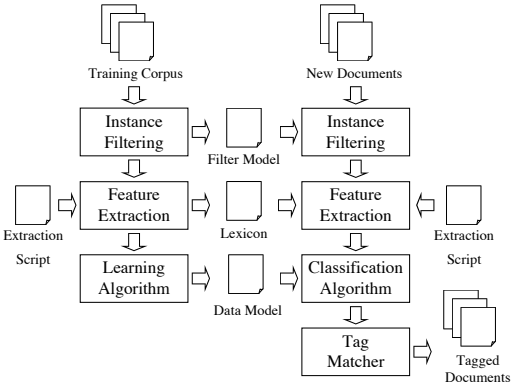

Figure 2: The SIE Architecture

The architecture of the system is represented in Figure 2. In the training phase, SIE learns off-line a set of data models from a corpus prepared in *IOBE* format (see 4.1). In the classification phase, these models are applied to tag new documents. In both the phases, the IF module is used to remove instances from the data sets, and the Feature Extraction module is applied to all the unfiltered instances. Each entity boundary is independently identified by the Classification Module, and the final output is provided by the Tag Matcher module. In the following subsections we will briefly describe each module.

[6]We implemented an algorithm for finding the optimal threshold with logarithmic time complexity.

4.1 Input Format

The corpus must be prepared in *IOBE* notation, an *ad-hoc* extension of the *IOB* notation. Both notations do not allow nested and overlapping entities. Tokens outside entities are tagged with **O**, the first token of an entity is tagged with **B-entity_type**, the last token is tagged **E-entity_type**, and all the tokens inside the entity boundaries are tagged with **I-entity_type**, where **entity_type** is the type of the marked entity (e.g. *protein*, *person*).

4.2 Instance Filtering Module

The IF module implements the three different Stop Word Filters described in Section 3.2[7]. Two different Stop Word Lists are provided for the beginning and the end boundaries of each entity, as SIE learns two distinct classifiers for them.

4.3 Feature Extraction

The Feature Extraction module is used to extract a predefined set of features for each unfiltered token in both the training and the test sets. Each instance is represented by encoding all the following basic features for the token itself and for all the tokens in a context window of size ± 3 tokens:

Type The type of the token.

POS The Part of Speech of the token.

Orthographic These features map each token into equivalence classes that encode attributes such as *capitalization*, *upper-case*, *numeric*, *single character*, and *punctuation*.

Moreover, the Feature Extraction module encodes all the bigrams of tokens and PoSs in a local window.

4.4 Classification

SIE approaches the IE task as a classification problem by assigning an appropriate classification label to unfiltered tokens. We use SVM^{light} for training the classifiers[8]. In particular, SIE identifies the *boundaries* that indicate the beginning and the end of each entity as two distinct classification tasks, following the approach adopted in [3; 6]. All tokens that begin(end) an entity are considered positive instances for the begin(end) classifier, while all the remaining tokens are negative instances. In this way, SIE uses $2n$ binary classifiers, where n is the number of entity_types for the task.

4.5 Tag Matcher

All the positive predictions produced by the begin and end classifiers are paired by the Tag Matcher module, that provides the final output of the system. To perform this operation, the Tag Matcher module assigns a score to each candidate entity. If nested or overlapping entities occur, it selects the entity with the maximal score.

The score of each entity is proportional to the entity length probability (i.e. the probability that an entity has a certain length) and to the confidence provided by the classifiers to the boundary predictions. The entity length distribution is estimated from the training set, as reported in Table 2.

[7]All the IF techniques described in this paper are implemented in the java Instance Filtering tool (jInFil), freely available at `http://tcc.itc.it/research/textec/tools-resources/jinfil.html`.

[8]SVM^{light} is available at `http://svmlight.joachims.org/`.

Table 1: A corpus fragment with multiple predictions. The left column shows the actual label, while the right column shows the predictions and their normalized scores

O	interact	
O	with	
O	the	
O	functional	
B-protein	kappa	`B-protein (0.23)`
I-protein	B	`B-protein(0.1), E-protein (0.12)`
I-protein	enhancer	`E-protein (0.34)`
O	present	
O	in	
O	the	

For example, in the corpus fragment reported in Table 1, the Classification Module have identified four possible boundaries for the entity `protein`. The matching algorithm chooses among three mutually exclusive candidates: "kappa B enhancer", "kappa B", and "B enhancer". The scores for each candidate are $0.23 \times 0.12 \times 0.33 = 0.009108$, $0.23 \times 0.34 \times 0.28 = 0.021896$ and $0.1 \times 0.34 \times 0.33 = 0.01122$, respectively. In the example, the matcher correctly extracts the candidate that maximizes the score function.

Table 2: The length distribution for the entity `protein` in the JNLPBA task

entity length	1	2	3	4	5	...
P(entity length)	0.10	0.33	0.28	0.07	0.02	...

5. EVALUATION

In order to assess the portability and the language independence of our filtering techniques, we performed a set of comparative experiments on three different tasks in two different languages (see Subsection 5.1). SIE exploited exactly the same configuration for all the tasks. We report results for three different Stop Word Filters: Information Content (IC), Odds Ratio (OR) and Correlation Coefficient (CC). In Subsection 5.2 we report the filtering rates obtained by varying the parameter ϵ[9]. The results show that IF drastically decreases the computation time (see Subsection 5.3), while preserving or improving the overall accuracy of the system (see Subsection 5.4).

5.1 Task Descriptions

The experiments reported in this paper have been performed in the following evaluation tasks:

JNLPBA The *JNLPBA* shared task [10] is an open challenge task proposed at the *"International Joint Workshop on Natural Language Processing in Biomedicine and its Application"*[10]. The data set consists of 2,404 MEDLINE abstracts from the GENIA project [9]. The abstracts were annotated with five entity_types: `DNA`, `RNA`, `protein`, `cell-line`, and `cell-type`.
The *JNLPBA* task splits the GENIA corpus into two partitions: the training partition consists of 492,551 tokens, the test partition consists of 101,039 tokens. The fraction of positive examples with respect to the total number of tokens in the training set varies from about 0.2% to about 6%.

CoNLL-2002 The CoNLL-2002 shared task is to recognize named entities from Dutch texts[11]. Four types of named entities are considered: persons, locations, organizations and names of miscellaneous entities that do not belong to the previous three groups. The Dutch corpus is divided into three partitions: the training partition and the validation partition, that on the whole consist of 258,214 tokens, and the test partition, containing 73,866 tokens. The fraction of positive examples with respect to the total number of tokens in the training set varies from about 1.1% to about 2%.

TERN The TERN (Time Expression Recognition and Normalization) 2004 Evaluation[12] requires systems to detect and normalize temporal expressions occurring in English text (SIE did not address the normalization part of the task). The TERN corpus is divided into two partitions: the training partition consists of 249,295 tokens and the test partition consists of 72,667 tokens. The fraction of positive examples with respect to the total number of tokens in the training set is about 2.1%.

5.2 Filtering Rates

Figure 3 displays the averaged filtering rates in the training and test sets on *JNLPBA*, *CoNLL-2002* and *TERN* using IC, CC and OR filters, respectively. The results indicate that both CC and OR do exhibit good performance and are far better than IC in all the tasks. For example, in the *JNLPBA* data set, OR allows to remove more than 70% of the instances, losing less than 1% of the positive examples. These results pinpoint the importance of using a supervised metric to collect stop words. The results also highlight that our optimization strategy is robust against overfitting, because the difference between the filtering rates in the training and test sets is minimal, satisfying the requirement expressed by equation 4. We also report a significant reduction of the data skewness. Table 3 shows that all the IF techniques reduce sensibly the data skewness on the JNLPBA data set[13]. As expected, both CC and OR consistently outperform IC.

5.3 Time Reduction

Figure 4 displays the impact of IF on the computation time[14] required to perform the overall IE process. The computation time includes both IF optimization and training and testing the boundary classifiers for each entity. It is important to note that the cost of the IF optimization process is negligible: about 230, 128 and 32 seconds for the *JNLPBA*, *CoNLL-2002* and *TERN* shared tasks, respectively. The curves indicate that both CC and OR are far superior to IC, allowing a drastic reduction of the time. Supervised

[9] We tried $\epsilon = 0, 0.01, 0.025, 0.05$. $\epsilon = 0$ means that the original data set has not been filtered.

[10] `http://research.nii.ac.jp/~collier/workshops/JNLPBA04st.htm`.

[11] `http://cnts.uia.ac.be/signll/shared.html`.

[12] `http://timex2.mitre.org/tern.html`.

[13] We only report results for this data set as it exhibits the highest skewness ratio.

[14] All the experiments have been performed using a dual 1.66 GHz Power Mac G5.

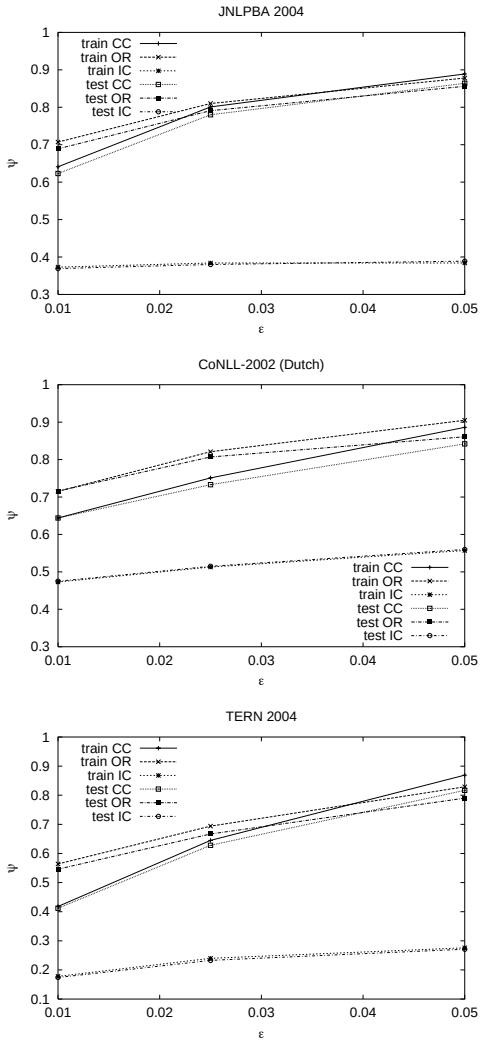

Figure 3: Averaged Filtering Rates (ψ) compared on training and test sets for *JNLPBA*, *CoNLL-2002* and *TERN*

IF techniques are then particularly convenient when dealing with large data sets. For example, the time required by SIE to perform the *JNLPBA* task decreased from more than 600 minutes to about 150 minutes (see Table 4).

5.4 Prediction Accuracy

Figure 5 plots the values of the micro-averaged F_1[15]. Both OR and CC allows to drastically reduce the computation time and maintain the prediction accuracy with small values of ϵ. Using OR, for example, with $\epsilon = 0.025$ on *JNLPBA*, F_1 augments from 66.7% to 67.9%. On the contrary, for *CoNLL-2002* and *TERN*, for $\epsilon > 0.025$ and $\epsilon > 0.01$ respectively, the performance of all the filters rapidly declines. The explanation for this behavior is that, for the last two tasks, the difference between the filtering rates on the training and test sets becomes much larger for $\epsilon > 0.025$ and $\epsilon > 0.01$, respectively. That is, the data skewness changes significantly from the training to the test set. It is not surprising that an extremely aggressive filtering step reduces too much the information available to the classifiers, leading the overall performance to decrease.

[15]All results are obtained using the official evaluation software made available by the organizers of the tasks.

Table 3: Skewness ratio of each entity for *JNLPBA*

entity	ϵ	CC	OR	IC
protein	0	17.1	17.1	17.1
	0.01	7.5	3.8	9.6
	0.025	3.0	2.5	9.0
	0.05	1.5	1.4	8.8
DNA	0	59.3	59.3	59.3
	0.01	26.4	18.5	33.2
	0.025	14.7	12.6	31.7
	0.05	8.3	8.6	32.4
RNA	0	596.2	596.2	596.2
	0.01	250.7	253.1	288.4
	0.025	170.4	170.1	274.5
	0.05	92.4	111.1	280.7
cell_type	0	72.9	72.9	72.9
	0.01	13.8	13.4	43.2
	0.025	6.3	6.5	43.9
	0.05	3.4	4.4	44.5
cell_line	0	146.4	146.4	146.4
	0.01	40.4	41.6	87.7
	0.025	24.2	25.9	87.5
	0.05	13.6	14.6	89.6

Table 4: Filtering Rate, Micro-averaged Recall, Precision, F_1 and Time for *JNLPBA*

Metric	ϵ	$\psi_{L/T}$	R	P	F_1	T
	0		0.664	0.670	0.667	615
CC	0.01	0.641/0.623	0.675	0.673	0.674	420
	0.025	0.801/0.780	0.666	**0.691**	0.678	226
	0.05	0.889/0.864	0.648	0.681	0.664	109
OR	0.01	0.707/0.689	**0.683**	0.673	0.678	308
	0.025	0.810/0.791	0.675	0.683	**0.679**	193
	0.05	0.878/0.856	0.654	0.682	0.668	114
IC	0.01	0.373/0.369	0.585	0.657	0.619	570
	0.025	0.384/0.380	0.569	0.654	0.609	558
	0.05	0.395/0.389	0.556	0.655	0.601	552
Zhou & Su [22]			0.760	0.694	0.726	
baseline			0.526	0.436	0.477	

5.5 Comparison with the state-of-the-art

Tables 4, 5 and 6 summarize the performance of SIE compared to the baselines and to the best systems in all the tasks. SIE largely outperforms all the baselines, achieving performances close to the best systems in all the tasks[16]. It is worth noting that state-of-the-art IE systems exploit external information (e.g. gazeteers [1] and lexical resources [22]) while SIE adopts exactly the same feature set and does not use any external or task dependent knowledge source.

6. CONCLUSION AND FUTURE WORK

The original motivation of our work was to approach the entity recognition task by using string kernels in a support vector machine based learning system. The high complexity of these algorithms suggested us to investigate in the direction of Instance Filtering as a preprocessing technique to alleviate two relevant problems of classification-based learning: the skewness of the class distribution and the data set

[16]Note that the TERN results cannot be publicly compared.

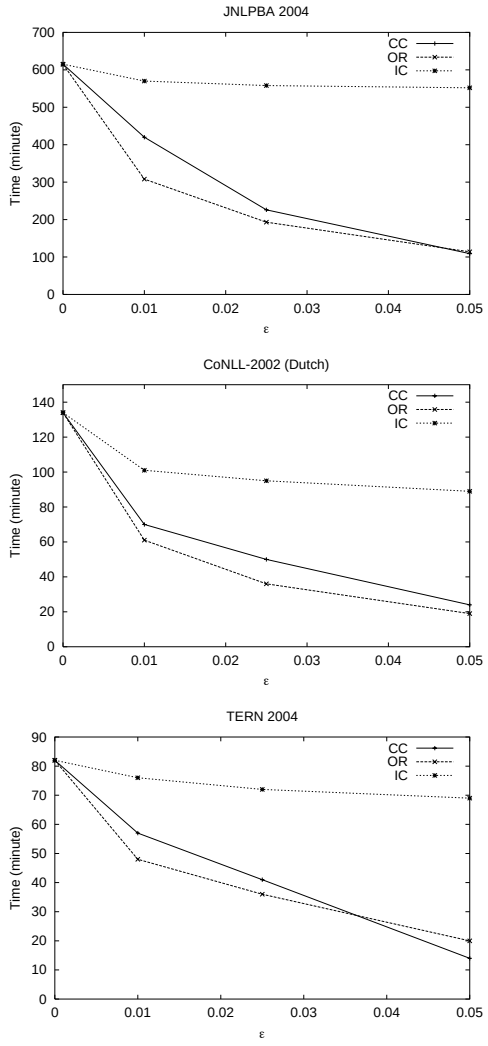

Figure 4: Computation time required by the SIE system to perform the overall IE process for *JNLPBA*, *CoNLL-2002* and *TERN*

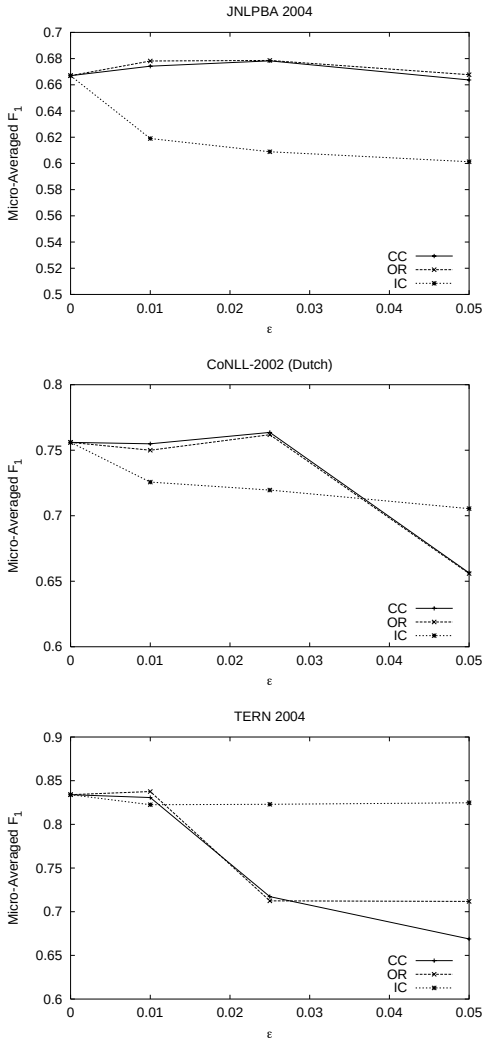

Figure 5: Micro-Averaged F_1 values for *JNLPBA*, *CoNLL-2002* and *TERN*

size. An important advantage of Instance Filtering is the drastic reduction of the computation time required by the entity recognition system to perform both training and classification. We presented a class of instance filters, namely Stop Word Filters, based on feature selection metrics. We did an extensive set of comparative experiments on different tasks in different languages, using a set of Instance Filtering techniques as preprocessing modules for the SIE system. In all the experiments we performed, the results are close to the state-of-the-art.

The portability, the language independence and the efficiency of SIE suggest its applicability in practical problems (e.g. user modeling, semantic web, information extraction from biological data) in which huge collections of texts have to be processed in a very limited time. In addition, the improved efficiency will allow us to experiment with kernel methods. For the future, we plan to implement more aggressive instance filtering schemata for Entity Recognition, by performing a deeper semantic analysis of the texts.

7. ACKNOWLEDGMENTS

Claudio Giuliano is supported by the IST-Dot.Kom project sponsored by the European Commission (Framework V grant IST-2001-34038). Raffaella Rinaldi is supported by the Web-FAQ project, funded by the Provincia Autonoma di Trento. We are grateful to Bruno Caprile, Cesare Furlanello and Alberto Lavelli for the helpful suggestions and comments. Thanks to Bonaventura Coppola for the preprocessing of CoNLL data set.

8. REFERENCES

[1] X. Carreras, L. Márques, and L. Padró. Named entity extraction using AdaBoost. In *Proceedings of CoNLL-2002*, Taipei, Taiwan, 2002.

[2] N. V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*, 6(1):1–6, 2004.

Table 5: Filtering Rate, Micro-averaged Recall, Precision, F_1 and total computation time for *CoNLL-2002*

Metric	ϵ	$\psi_{L/T}$	R	P	F_1	T
	0		0.736	0.787	0.761	134
CC	0.01	0.644/0.644	0.716	0.799	0.755	70
	0.025	0.751/0.733	0.728	**0.803**	**0.764**	50
	0.05	0.886/0.842	0.666	0.647	0.656	24
OR	0.01	0.715/0.716	0.720	0.783	0.750	61
	0.025	0.821/0.807	**0.736**	0.789	0.762	39
	0.05	0.905/0.861	0.668	0.645	0.656	19
IC	0.01	0.473/0.475	0.670	0.792	0.726	101
	0.025	0.513/0.515	0.659	0.793	0.720	95
	0.05	0.557/0.560	0.638	0.789	0.705	89
Carreras et al. [1]			0.763	0.778	0.771	
baseline			0.454	0.813	0.583	

Table 6: Filtering Rate, Micro-averaged Recall, Precision, F_1 and total computation time for *TERN*

Metric	ϵ	$\psi_{L/T}$	R	P	F_1	T
	0		0.779	0.898	0.834	82
CC	0.01	0.418/0.412	0.766	0.907	0.831	57
	0.025	0.645/0.628	0.603	0.886	0.717	41
	0.05	0.869/0.817	0.597	0.760	0.669	14
OR	0.01	0.564/0.546	**0.775**	0.911	**0.838**	48
	0.025	0.694/0.667	0.598	0.881	0.712	36
	0.05	0.829/0.790	0.595	0.886	0.712	20
IC	0.01	0.178/0.174	0.749	0.912	0.823	48
	0.025	0.240/0.233	0.748	**0.915**	0.823	36
	0.05	0.276/0.271	0.750	**0.915**	0.825	20

[3] F. Ciravegna. Learning to tag for information extraction. In F. Ciravegna, R. Basili, and R. Gaizauskas, editors, *Proceedings of the ECAI workshop on Machine Learning for Information Extraction*, Berlin, 2000.

[4] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[5] A. De Sitter and W. Daelemans. Information extraction via double classification. In *International Workshop on Adaptive Text Extraction and Mining*, 2003.

[6] D. Freitag and N. Kushmerick. Boosted wrapper induction. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI 2000)*, pages 577–583, 2000.

[7] C. Giuliano, A. Lavelli, and L. Romano. Simple information extraction (SIE). Technical report, ITC-irst, Istituto per la Ricerca Scientifica e Tecnologica, 2005.

[8] A. M. Gliozzo, C. Giuliano, and R. Rinaldi. Instance pruning by filtering uninformative words: an Information Extraction case study. In *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2005)*, Mexico City, Mexico, 13-19 February 2005.

[9] J. Kim, T. Ohta, Y. Tateishi, and J. Tsujii. Genia corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(Suppl.1):180–182, 2003.

[10] J. Kim, T. Ohta, Y. Tsuruoka, Y. Tateisi, and N. Collier. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA-2004)*, Geneva, Switzerland, 2004.

[11] S. Kotsiantis and P. Pintelas. Mixture of expert agents for handling imbalanced data sets. *Annals of Mathematics, Computing and Teleinformatics*, 1(1):46–55, 2003.

[12] J. Leskovec and J. Shawe-Taylor. Linear programming boosting for uneven datasets. In T. Fawcett and N. Mishra, editors, *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 456–463. AAI Press, 2003.

[13] D. Roth and W. Yih. Relational learning via propositional algorithms: An information extraction case study. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, 2001.

[14] Y. Song, E. Yi, E. Kim, and G. G. Lee. POSBIOTM-NER in the shared task of bionlp/nlpba2004. In *Proceedings of the Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA-2004)*, Geneva, Switzerland, 2004.

[15] I. Steinwart. Sparseness of Support Vector Machines—some asymptotically sharp bounds. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[16] G. Weiss and F. Provost. The effect of class distribution on classifier learning. Technical Report ML-TR 43, Department of Computer Science, Rutgers University, 2001.

[17] G. M. Weiss. Mining with rarity: a unifying framework. *SIGKDD Explorations*, 6(1):7–19, 2004.

[18] D. R. Wilson and T. R. Martinez. Instance pruning techniques. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 403–411, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[19] D. R. Wilson and T. R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38(3):257–286, 2000.

[20] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In D. H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[21] Z. Zheng, X. Wu, and R. Srihari. Feature selection for text categorization on imbalanced data. *SIGKDD Explor. Newsl.*, 6(1):80–89, 2004.

[22] G. D. Zhou and J. Su. Exploring deep knowledge resources in biomedical name recognition. In *Proceedings of the Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA-2004)*, Geneva, Switzerland, 2004.