Simple Information Extraction (SIE): A Portable and Effective IE System

Claudio Giuliano and **Alberto Lavelli** and **Lorenza Romano**
ITC-irst
Via Sommarive, 18
38050, Povo (TN)
Italy
{giuliano,lavelli,romano}@itc.it

Abstract

This paper describes SIE (Simple Information Extraction), a modular information extraction system designed with the goal of being easily and quickly portable across tasks and domains. SIE is composed by a general purpose machine learning algorithm (SVM) combined with several customizable modules. A crucial role in the architecture is played by Instance Filtering, which allows to increase efficiency without reducing effectiveness. The results obtained by SIE on several standard data sets, representative of different tasks and domains, are reported. The experiments show that SIE achieves performance close to the best systems in all tasks, without using domain-specific knowledge.

1 Introduction

In designing Information Extraction (IE) systems based on supervised machine learning techniques, there is usually a tradeoff between carefully tuning the system to specific tasks and domains and having a ”generic” IE system able to obtain good (even if not the topmost) performance when applied to different tasks and domains (requiring a very reduced porting time). Usually, the former alternative is chosen and system performance is often shown only for a very limited number of tasks (sometimes even only for a single task), after a careful tuning. For example, in the Bio-entity Recognition Shared Task at JNLPBA 2004 (Kim et al., 2004) the best performing system obtained a considerable performance improvement adopting domain specific hacks.

A second important issue in designing IE systems concerns the fact that usually IE data sets are highly unbalanced (i.e., the number of positive examples constitutes only a small fraction with respect to the number of negative examples). This fact has important consequences. In some machine learning algorithms the unbalanced distribution of examples can yield a significant loss in classification accuracy. Moreover, very large data sets can be problematic to process due to the complexity of many supervised learning techniques. For example, using kernel methods, such as word sequence and tree kernels, can become prohibitive due to the difficulty of kernel based algorithms, such as Support Vector Machines (SVM) (Cortes and Vapnik, 1995), to scale to large data sets. As a consequence, reducing the number of instances without degrading the prediction accuracy is a crucial issue for applying advanced machine learning techniques in IE, especially in the case of highly unbalanced data sets.

In this paper, we present SIE (Simple Information Extraction), an information extraction system based on a supervised machine learning approach for extracting domain-specific entities from documents. In particular, IE is cast as a classification problem by applying SVM to train a set of classifiers, based on a simple and general-purpose feature representation, for detecting the boundaries of the entities to be extracted.

SIE was designed with the goal of being easily and quickly portable across tasks and domains. To support this claim, we conducted a set of experiments on several tasks in different domains and languages. The results show that SIE is competitive with the state-of-the-art systems, and it often outperforms systems customized to a specific domain.

SIE resembles the “Level One” of the ELIE algorithm (Finn and Kushmerick, 2004). However, a key difference between the two algorithms is the capability of SIE to drastically reduce the computation time by exploiting Instance Filtering (Gliozzo et al., 2005a). This characteristic allows scaling from toy problems to real-world data sets making SIE attractive in applicative fields, such as bioinformatics, where very large amounts of data have to be analyzed.

2 A Simple IE system

SIE has a modular system architecture. It is composed by a general purpose machine learning algorithm combined with several customizable components. The system components are combined in a pipeline, where each module constrains the data structures provided by the previous ones. This modular specification brings significant advantages. Firstly, a modular architecture is simpler to implement. Secondly, it allows to easily integrate different machine learning algorithms. Finally, it allows, if necessary, a fine tuning to a specific task by simply specializing few modules. Furthermore, it is worth noting that we tested SIE across different domains using the same basic configuration without exploiting any domain specific knowledge, such as gazetteers, and *ad-hoc* pre/post-processing.

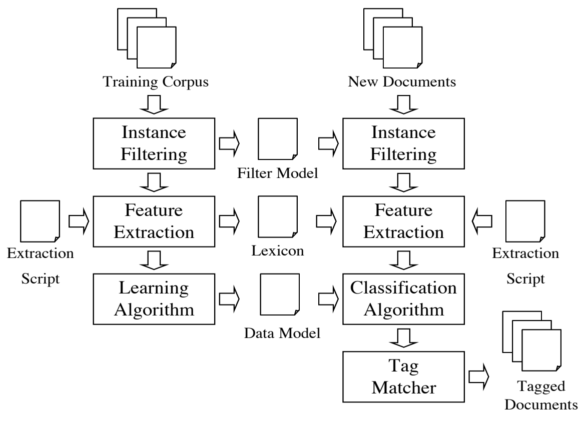

Figure 1: The SIE Architecture.

The architecture of the system is shown in Figure 1. The information extraction task is performed in two phases. SIE learns off-line a set of data models from a specified labeled corpus, then the models are applied to tag new documents.

In both phases, the Instance Filtering module (Section 3) removes certain tokens from the data set in order to speed-up the whole process, while Feature Extraction module (Section 4) is used to extract a pre-defined set of features from the tokens. In the training phase, the Learning Module (Section 5) learns two distinct models for each entity, one for the beginning boundary and another for the end boundary (Ciravegna, 2000; Freitag and Kushmerick, 2000). In the recognition phase, as a consequence, the Classification module (Section 5) identifies the entity boundaries as distinct token classifications. A Tag Matcher module (Section 6) is used to match the boundary predictions made by the Classification module. Tasks with multiple entities are considered as multiple independent single-entity extraction tasks (i.e. SIE only extracts one entity at a time).

3 Instance Filtering

The purpose of the Instance Filtering (IF) module is to reduce the data set size and skewness by discarding harmful and superfluous instances without degrading the prediction accuracy. This is a generic module that can be exploited by any supervised system that casts IE as a classification problem.

Instance Filtering (Gliozzo et al., 2005a) is based on the assumption that uninformative words are not likely to belong to entities to recognize, being their information content very low. A naive implementation of this assumption consists in filtering out very frequent words in corpora because they are less likely to be relevant than rare words. However, in IE relevant entities can be composed by more than one token and in some domains a few of such tokens can be very frequent in the corpus. For example, in the field of bioinformatics, protein names often contain parentheses, whose frequency in the corpus is very high.

To deal with this problem, we exploit a set of Instance Filters (called *Stop Word Filters*), included in a Java tool called jInFil[1]. These filters perform a “shallow” supervision to identify frequent words that are often marked as positive examples. The resulting filtering algorithm consists of two stages. First, the set of uninformative tokens is identified by training the term filtering algorithm on the training corpus. Second, instances describing “uninformative” tokens are removed from both the training and the test sets. Note that *instances* are not really removed from the data set, but just

[1]`http://tcc.itc.it/research/textec/tools-resources/jinfil/`

marked as uninformative. In this way the learning algorithm will not learn from these instances, but they will still appear in the feature description of the remaining instances.

A Stop Word Filter is fully specified by a list of stop words. To identify such a list, different feature selection methods taken from the text categorization literature can be exploited. In text categorization, feature selection is used to remove non-informative terms from 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 describe the different metrics used to collect a stop word list from the training corpora.

Information Content (IC) The most commonly used feature selection metric in text categorization is based on document frequency (i.e, the number of documents in which a term occurs). The basic assumption is that very frequent terms are non-informative for document indexing. The frequency of a term in the corpus is a good indicator of its generality, rather than of its information content. From this point of view, IF consists of removing all tokens with a very low information content[2].

Correlation Coefficient (CC) In text categorization the χ^2 statistic is used to measure the lack of independence between a term and a category (Yang and Pedersen, 1997). The correlation coefficient $CC^2 = \chi^2$ of a term with the negative class can be used to find those terms that are less likely to express relevant information in texts.

Odds Ratio (OR) Odds ratio measures the ratio between the odds of a term occurring in the positive class, and the odds of a term occurring in the negative class. In text categorization the idea is that the distribution of the features on the relevant documents is different from the distribution on non-relevant documents (Raskutti and Kowalczyk, 2004). Following this assumption, a term is non-informative when its probability of being a negative example is sensibly higher than its probability of being a positive example (Gliozzo et al., 2005b).

[2]The information content of a word w can be measured by estimating its probability from a corpus by the equation $I(w) = -p(w) \log p(w)$.

An Instance Filter is evaluated by using two metrics: the Filtering Rate (ψ), the total percentage of filtered tokens in the data set, and the Positive Filtering Rate (ψ^+), the percentage of positive tokens (wrongly) removed. A filter is optimized by maximizing ψ and minimizing ψ^+; this allows us to reduce as much as possible the data set size preserving most of the positive instances. We fixed the accepted level of tolerance (ϵ) on ψ^+ and found the maximum ψ by performing 5-fold cross-validation on the training set.

4 Feature Extraction

The Feature Extraction module is used to extract for each input token a pre-defined set of features. As said above, we consider each token an instance to be classified as a specific entity boundary or not. To perform Feature Extraction an application called jFex[3] was implemented. jFex generates the features specified by a feature extraction script, indexes them, and returns the example set, as well as the mapping between the features and their indices (lexicon). If specified, it only extracts features for the instances not marked as "uninformative" by instance filtering. jFex is strongly inspired by FEX (Cumby and Yih, 2003), but it introduces several improvements. First of all, it provides an enriched feature extraction language. Secondly, it makes possible to further extend this language through a Java API, providing a flexible tool to define task specific features. Finally, jFex can output the example set in formats directly usable by LIBSVM (Chang and Lin, 2001), SVM^{light} (Joachims, 1998) and SNoW (Carlson et al., 1999).

4.1 Corpus Format

The corpus must be prepared in *IOBE* notation, a extension of the *IOB* notation. Both notations do not allow nested and overlapping entities. Tokens outside entities are tagged with *O*, while 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).

Beside the tokens and their types, the notation allows to represent general purpose and task-specific annotations defining new columns. Blank

[3]`http://tcc.itc.it/research/textec/tools-resources/jfex.html`.

lines can be used to specify sentence or document boundaries. Table 1 shows an example of a prepared corpus. The columns are: the entity-type, the PoS tag, the actual token, the token index, and the output of the instance filter (the "uninformative" tokens are marked with 0) respectively.

O	TO	To	2.12	0
O	VB	investigate	2.13	0
O	IN	whether	2.14	0
O	DT	the	2.15	0
B-cell_type	NN	tumor	2.16	1
O	NN	expression	2.17	1
O	IN	of	2.18	0
B-protein	NN	Beta-2-Microglobulin	2.19	1
O	(	(	2.20	1
B-protein	NN	Beta	2.21	1
I-protein	NN	2	2.22	1
I-protein	NN	-	2.22	1
E-protein	NN	M	2.22	1
O	)	)	2.23	1

Table 1: A corpus fragment represented in *IOBE* notation.

4.2 Extraction Language

As input to the *begin* and *end* classifiers, we use a bit-vector representation. Each instance is represented encoding all the following basic features for the actual token and for all the tokens in a context window of fixed size (in the reported experiments, 3 words before and 3 words after the actual token):

Token The actual token.

POS The Part of Speech (PoS) of the token.

Token Shapes This feature maps each token into equivalence classes that encode attributes such as *capitalization*, *numerals*, *single character*, and so on.

Bigrams of tokens and PoS tags.

The Feature Extraction language allows to formally encode the above problem description through a script. Table 2 provides the extraction script used in all the tasks[4]. More details about the Extraction Language are provided in (Cumby and Yih, 2003; Giuliano et al., 2005).

[4]In JNLPBA shared task we added some orthographic features borrowed from the bioinformatics literature.

```
-1 inc loc: w [-3, 3]
-1 inc loc: coloc(w,w) [-3, 3]
-1 inc loc: t [-3, 3]
-1 inc loc: coloc(t,t) [-3, 3]
-1 inc loc: sh [-3, 3]
```

Table 2: The extraction script used in all tasks.

5 Learning and Classification Modules

As already said, we approach IE as a classification problem, assigning an appropriate classification label to each token in the data set except for the tokens marked as irrelevant by the instance filter. As learning algorithm we use SVM-light[5]. In particular, we identify the *boundaries* that indicate the beginning and the end of each entity as two distinct classification tasks, following the approach adopted in (Ciravegna, 2000; Freitag and Kushmerick, 2000). 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, two distinct models are learned, one for the beginning boundary and another for the end boundary. All the predictions produced by the *begin* and *end* classifiers are then paired by the Tag Matcher module.

When we have to deal with more than one entity (i.e., with a multi-class problem) we train $2n$ binary classifiers (where n is the number of *entity-types* for the task). Again, all the predictions are paired by the Tag Matcher module.

6 Tag Matcher

All the positive predictions produced by the *begin* and *end* classifiers are paired by the Tag Matcher module. If nested or overlapping entities occur, even if they are of different types, the entity with the highest score is selected. The score of each entity is proportional to the entity length probability (i.e., the probability that an entity has a certain length) and the scores assigned by the classifiers to the boundary predictions. Normalizing the scores makes it possible to consider the score function as a probability distribution. The entity length distribution is estimated from the training set.

For example, in the corpus fragment of Table 3 the *begin* and *end* classifiers have identified four possible entity boundaries for the speaker of a seminar. In the table, the left column shows the

[5]`http://svmlight.joachims.org/`

Table 3: A corpus fragment with multiple predictions.

O	The	
O	speaker	
O	will	
O	be	
B-speaker	Mr.	**B-speaker (0.23)**
I-speaker	John	**B-speaker (0.1), E-speaker (0.12)**
E-speaker	Smith	**E-speaker (0.34)**
O	.	

Table 4: The length distribution for the entity *speaker*.

entity len	1	2	3	4	5	...
P(entity len)	0.10	0.33	0.28	0.02	0.01	...

actual label, while the right column shows the predictions and their normalized scores. The matching algorithm has to choose among three mutually exclusive candidates: "Mr. John", "Mr. John Smith" and "John Smith", with scores $0.23 \times 0.12 \times 0.33 = 0.009108$, $0.23 \times 0.34 \times 0.28 = 0.021896$ and $0.1 \times 0.34 \times 0.33 = 0.01122$, respectively. The length distribution for the entity *speaker* is shown in Table 4. In this example, the matcher, choosing the candidate that maximizes the score function, namely the second one, extracts the actual entity.

7 Evaluation

In order to demonstrate that SIE is domain and language independent we tested it on several tasks using exactly the same configuration. The tasks and the experimental settings are described in Section 7.1. The results (Section 7.2) show that the adopted filtering technique decreases drastically the computation time while preserving (and sometimes improving) the overall accuracy of the system.

7.1 The Tasks

SIE was tested on the following IE benchmarks:

JNLPBA Shared Task This shared task (Kim et al., 2004) is an open challenge task proposed at the *"International Joint Workshop on Natural Language Processing in Biomedicine and its Applications"*[6]. The data set consists of 2,404 MEDLINE abstracts from the GENIA project (Kim et al., 2003), annotated with five entity types: *DNA*, *RNA*, *protein*, *cell-line*, and *cell-type*. The GENIA corpus is split into two partitions: training (492,551 tokens), and test (101,039 tokens). The fraction of positive examples with respect to the total number of tokens in the training set varies from 0.2% to 6%.

CoNLL 2002 & 2003 Shared Tasks These shared tasks (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003)[7] concern language-independent named entity recognition. Four types of named entities are considered: persons (*PER*), locations (*LOC*), organizations (*ORG*) and names of miscellaneous (*MISC*) entities that do not belong to the previous three groups. SIE was applied to the Dutch and English data sets. The Dutch corpus is divided into three partitions: training and validation (on the whole 258,214 tokens), and test (73,866 tokens). The fraction of positive examples with respect to the total number of tokens in the training set varies from 1.1% to 2%. The English corpus is divided into three partitions: training and validation (on the whole 274,585 tokens), and test (50,425 tokens). The fraction of positive examples with respect to the total number of tokens in the training set varies from 1.6% to 3.3%.

TERN 2004 The TERN (Time Expression Recognition and Normalization) 2004 Evaluation[8] 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: training (249,295 tokens) and test (72,667 tokens). The fraction of positive examples with respect to the total number of tokens in the training set is about 2.1%.

Seminar Announcements The Seminar Announcements (SA) collection (Freitag, 1998) consists of 485 electronic bulletin board postings. The purpose of each document in the collection is to announce or relate details of an upcoming talk or seminar. The documents were annotated for four entities: *speaker*, *location*, *stime*, and *etime*. The corpus is composed by 156,540 tokens. The fraction of positive examples varies from about 1% to

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

[7]http://www.cnts.ua.ac.be/conll2002/ner/, http://www.cnts.ua.ac.be/conll2003/ner/.

[8]http://timex2.mitre.org/tern.html.

Metric	ϵ	$\psi_{train/test}$	R	P	F_1	T
	0		66.4	67.0	66.7	615
CC	1	64.1/62.3	67.5	67.3	67.4	420
	2.5	80.1/78.0	66.6	**69.1**	67.8	226
	5	88.9/86.4	64.8	68.1	66.4	109
OR	1	70.7/68.9	**68.3**	67.3	67.8	308
	2.5	81.0/79.1	67.5	68.3	**67.9**	193
	5	87.8/85.6	65.4	68.2	66.8	114
IC	1	37.3/36.9	58.5	65.7	61.9	570
	2.5	38.4/38.0	56.9	65.4	60.9	558
	5	39.5/38.9	55.6	65.5	60.1	552
Zhou and Su (2004)			76.0	69.4	72.6	
baseline			52.6	43.6	47.7	

Table 5: Filtering Rate, Micro-averaged Recall, Precision, F_1 and Time for *JNLPBA*.

Metric	ϵ	$\psi_{train/test}$	R	P	F_1	T
	0		73.6	78.7	76.1	134
CC	1	64.4/64.4	71.6	79.9	75.5	70
	2.5	75.1/73.3	72.8	**80.3**	**76.4**	50
	5	88.6/84.2	66.6	64.7	65.6	24
OR	1	71.5/71.6	72.0	78.3	75.0	61
	2.5	82.1/80.7	**73.6**	78.9	76.2	39
	5	90.5/86.1	66.8	64.5	65.6	19
IC	1	47.3/47.5	67.0	79.2	72.6	101
	2.5	51.3/51.5	65.9	79.3	72.0	95
	5	55.7/56.0	63.8	78.9	70.5	89
Carreras et al. (2002)			76.3	77.8	77.1	
baseline			45.4	81.3	58.3	

Table 6: Filtering Rate, Micro-averaged Recall, Precision, F_1 and total computation time for *CoNLL-2002 (Dutch)*.

about 2%. The entire document collection is randomly partitioned five times into two sets of equal size, training and test (Lavelli et al., 2004). For each partition, learning is performed on the training set and performance is measured on the corresponding test set. The resulting figures are averaged over the five test partitions.

7.2 Results

The experimental results in terms of filtering rate, recall, precision, F_1, and computation time for *JNLPBA*, *CoNLL-2002*, *CoNLL-2003*, *TERN* and *SA* are given in Tables 5, 6, 7, 8 and 9 respectively. To show the differences among filtering strategies for *JNLPBA*, *CoNLL-2002*, *TERN 2004* we used CC, OR and IC filters, while the results for *SA* and *CoNLL-2003* are reported only for OR filter (which usually produces the best performance). For all filters we report results obtained by setting four different values for parameter ϵ, the maximum value allowed for the Filtering Rate of positive examples. $\epsilon = 0$ means that no filter is used.

Metric	ϵ	$\psi_{train/test}$	R	P	F_1	T
	0		76.7	**90.5**	**83.1**	228
OR	1	70.4/83.9	**78.2**	88.1	82.8	74
	2.5	83.6/95.6	76.4	62.6	68.8	33
	5	90.5/97.2	75.3	66.5	70.7	14
Florian et al. (2003)			88.5	89.0	88.8	
baseline			50.9	71.9	59.6	

Table 7: Filtering Rate, Micro-averaged Recall, Precision, F_1 and total computation time for *CoNLL-2003 (English)*.

Metric	ϵ	$\psi_{train/test}$	R	P	F_1	T
	0		**77.9**	89.8	83.4	82
CC	1	41.8/41.2	76.6	90.7	83.1	57
	2.5	64.5/62.8	60.3	88.6	71.7	41
	5	86.9/81.7	59.7	76.0	66.9	14
OR	1	56.4/54.6	77.5	91.1	**83.8**	48
	2.5	69.4/66.7	59.8	88.1	71.2	36
	5	82.9/79.0	59.5	88.6	71.2	20
IC	1	17.8/17.4	74.9	91.2	82.3	48
	2.5	24.0/23.3	74.8	**91.5**	82.3	36
	5	27.6/27.1	75.0	**91.5**	82.5	20

Table 8: Filtering Rate, Micro-averaged Recall, Precision, F_1 and total computation time for *TERN*.

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 both CC and OR are robust against overfitting, because the difference between the filtering rates in the training and test sets is minimal. We also report a significant reduction of the data skewness. Table 10 shows that all the IF techniques reduce sensibly the skewness ratio, the ratio between the number of negative and positive examples, on the JNLPBA data set[9]. As expected, both CC and OR consistently outperform IC.

The computation time[10] reported includes the time to perform the overall process of training and testing the boundary classifiers for each entity[11]. The results indicate that both CC and OR are far superior to IC, allowing a drastic reduction of the time. Supervised IF techniques are then particu-

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

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

[11]Execution time for filter optimization is not reported because it is negligible.

Metric	ϵ	$\psi_{train/test}$	R	P	F_1	T
	0		81.3	**92.5**	**86.6**	179
OR	1	53.6/86.2	81.5	92.1	86.5	91
	2.5	69.1/90.8	**81.6**	90.5	85.9	44
	5	74.7/90.8	81.0	85.0	83.0	31

Table 9: Filtering Rate, Micro-averaged Recall, Precision, F_1 and total computation time for *SA*.

entity	ϵ	CC	OR	IC
protein	0	17.1	17.1	17.1
	1	7.5	3.8	9.6
	2.5	3.0	2.5	9.0
	5	1.5	1.4	8.8
DNA	0	59.3	59.3	59.3
	1	26.4	18.5	33.2
	2.5	14.7	12.6	31.7
	5	8.3	8.6	32.4
RNA	0	596.2	596.2	596.2
	1	250.7	253.1	288.4
	2.5	170.4	170.1	274.5
	5	92.4	111.1	280.7
cell_type	0	72.9	72.9	72.9
	1	13.8	13.4	43.2
	2.5	6.3	6.5	43.9
	5	3.4	4.4	44.5
cell_line	0	146.4	146.4	146.4
	1	40.4	41.6	87.7
	2.5	24.2	25.9	87.5
	5	13.6	14.6	89.6

Table 10: Skewness ratio of each entity for *JNLPBA*.

larly convenient when dealing with large data sets. For example, using the CC metric the time required by SIE to perform the *JNLPBA* task is reduced from 615 to 109 minutes (see Table 5).

Both OR and CC allow to drastically reduce the computation time and maintain the prediction accuracy[12] with small values of ϵ. Using OR, for example, with $\epsilon = 2.5\%$ on *JNLPBA*, F_1 increases from 66.7% to 67.9%. On the contrary, for *CoNLL-2002* and *TERN*, for $\epsilon > 2.5\%$ and $\epsilon > 1\%$ 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 $\epsilon > 2.5\%$ and $\epsilon > 1\%$, 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.

SIE achieves results close to the best systems in all tasks[13]. It is worth noting that state-of-the-art IE systems often exploit external, domain-specific information (e.g. gazetteers (Carreras et al., 2002) and lexical resources (Zhou and Su, 2004)) while SIE adopts exactly the same feature set and does not use any external or task dependent knowledge source.

[12]For JNLPBA, CoNLL 2002 & 2003 and Tern 2004, results are obtained using the official evaluation software made available by the organizers of the tasks.

8 Conclusion and Future Work

The portability, the language independence and the efficiency of SIE suggest its applicability in practical problems (e.g. semantic web, information extraction from biological data) in which huge collections of texts have to be processed efficiently. In this perspective we are pursuing the recognition of bio-entities from several thousands of MEDLINE abstracts. In addition, the effectiveness of instance filtering will allow us to experiment with complex 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.

Acknowledgments

SIE was developed in the context of the IST-Dot.Kom project (`http://www.dot-kom.org`), sponsored by the European Commission as part of the Framework V (grant IST-2001-34038). Claudio Giuliano and Lorenza Romano have been supported by the ONTOTEXT project, funded by the Autonomous Province of Trento under the FUP-2004 research program.

References

Andrew J. Carlson, Chad M. Cumby, Jeff L. Rosen, and Dan Roth. 1999. SNoW user's guide. Technical Report UIUCDCS-DCS-R-99-210, Department of Computer Science, University of Illinois at Urbana-Champaign, April.

Xavier Carreras, Lluís Márques, and Lluís Padró. 2002. Named entity extraction using adaboost. In *Proceedings of CoNLL-2002*, Taipei, Taiwan.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines.*

[13]Note that the TERN results cannot be disclosed, so no direct comparison can be provided. For the reasons mentioned in (Lavelli et al., 2004), direct comparison cannot be provided for Seminar Announcements as well.

Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Fabio Ciravegna. 2000. 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.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

Chad Cumby and W. Yih. 2003. FEX user guide. Technical report, Department of Computer Science, University of Illinois at Urbana-Champaign, April.

Aidan Finn and Nicholas Kushmerick. 2004. Multi-level boundary classification for information extraction. In *Proceedings of the 15th European Conference on Machine Learning*, Pisa, Italy.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 168–171. Edmonton, Canada.

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

Dayne Freitag. 1998. *Machine Learning for Information Extraction in Informal Domains*. Ph.D. thesis, Carnegie Mellon University.

Claudio Giuliano, Alberto Lavelli, and Lorenza Romano. 2005. Simple information extraction (SIE). Technical report, ITC-irst.

Alfio Massimiliano Gliozzo, Claudio Giuliano, and Raffaella Rinaldi. 2005a. Instance filtering for entity recognition. *SIGKDD Explorations (special issue on Text Mining and Natural Language Processing)*, 7(1):11–18, June.

Alfio Massimiliano Gliozzo, Claudio Giuliano, and Raffaella Rinaldi. 2005b. 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.

T. Joachims. 1998. Making large-scale support vector machine learning practical. In A. Smola B. Schölkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA.

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

J. Kim, T. Ohta, Y. Tsuruoka, Y. Tateisi, and N. Collier. 2004. Introduction to the bio-entity recognition task at JNLPBA. In N. Collier, P. Ruch, and A. Nazarenko, editors, *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA-2004)*, pages 70–75, Geneva, Switzerland, August 28–29.

A. Lavelli, M. Califf, F. Ciravegna, D. Freitag, C. Giuliano, N. Kushmerick, and L. Romano. 2004. IE evaluation: Criticisms and recommendations. In *AAAI-04 Workshop on Adaptive Text Extraction and Mining (ATEM-2004)*, San Jose, California.

Bhavani Raskutti and Adam Kowalczyk. 2004. Extreme re-balancing for SVMs: a case study. *SIGKDD Explor. Newsl.*, 6(1):60–69.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. 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.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002*, pages 155–158. Taipei, Taiwan.

Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In Douglas H. Fisher, editor, *Proceedings of the 14th International Conference on Machine Learning (ICML-97)*, pages 412–420, Nashville, US. Morgan Kaufmann Publishers, San Francisco, US.

Guo Dong Zhou and Jian Su. 2004. Exploring deep knowledge resources in biomedical name recognition. In *Proceedings of 2004 Joint Workshop on Natural Processing in Biomedicine and its Applications*, Geneva, Switzerland.