Adaptive Information Extraction from Text by Rule Induction and Generalisation

Fabio Ciravegna
Department of Computer Science, University of Sheffield
Regent Court, 211 Portobello Street,
S1 4DP Sheffield, UK
F.Ciravegna@dcs.shef.ac.uk

Abstract

$(LP)^2$ is a covering algorithm for adaptive Information Extraction from text (IE). It induces symbolic rules that insert SGML tags into texts by learning from examples found in a user-defined tagged corpus. Training is performed in two steps: initially a set of tagging rules is learned; then additional rules are induced to correct mistakes and imprecision in tagging. Induction is performed by bottom-up generalization of examples in the training corpus. Shallow knowledge about Natural Language Processing (NLP) is used in the generalization process. The algorithm has a considerable success story. From a scientific point of view, experiments report excellent results with respect to the current state of the art on two publicly available corpora. From an application point of view, a successful industrial IE tool has been based on $(LP)^2$. Real world applications have been developed and licenses have been released to external companies for building other applications. This paper presents $(LP)^2$, experimental results and applications, and discusses the role of shallow NLP in rule induction.

1. Introduction

By general agreement the main barriers to wide use and commercialization of Information Extraction from text (IE) are the difficulties in adapting systems to new applications. The classical IE has been focusing on applications to free texts; therefore systems often rely on approaches based on Natural Language Processing (NLP) (e.g. using parsing) [Humphreys *et al.* 1997; Grishman 1997]. Most systems require the manual development of resources (e.g. grammars) by a user skilled in NLP [Ciravegna 2000]. There is an increasing interest in applying machine learning (ML) to IE in order to build adaptive systems. Up to now, the use of ML has been approached mainly in an NLP-oriented perspective, i.e. in order to reduce the amount of work to be done by the NLP experts in porting systems across free text based scenarios [Cardie 1997; Miller *et al.* 1998; Yangarber *et al.* 2000]. Given the current technology, IE experts are still necessary.

In the last years, the increasing importance of the Internet has stressed the central role of texts such as emails, Usenet posts and Web pages. In this context, extralinguistic structures (e.g. HTML tags, document formatting, and ungrammatical stereotypical language) are elements used to convey information. Linguistically intensive approaches are difficult or unnecessary in such cases. For this reason a new research stream on adaptive IE has arisen at the convergence of NLP, Information Integration and Machine Learning. The goal is to produce IE algorithms and systems adaptable to new Internet-related applications/scenarios by using only an analyst's knowledge (i.e. knowledge on the domain/scenario itself) [Kushmerick 1997; Califf 1998; Muslea *et al.* 1998; Freitag and McCallum 1999; Soderland 1999; Freitag and Kushmerick 2000]. Such algorithms are very effective when applied on highly structured HTML pages, but less effective on unstructured texts (e.g. free texts). In our opinion this is because most successful algorithms make scarce (or no) use of NLP, tending to avoid any generalization over the flat word sequence. When they are applied to unstructured texts, data sparseness becomes a problem.

This paper presents $(LP)^2$, an adaptive IE algorithm designed in this new stream of research that makes use of shallow NLP in order to overcome data sparseness when confronted with NL texts, while keeping effectiveness on highly structured texts. This paper first introduces the algorithm, discusses experimental results and shows how the algorithm compares successfully with the current state of the art. The role and importance of shallow NLP for overcoming data sparseness is then discussed. Finally a successful industrial system for adaptive IE built around $(LP)^2$ is presented and some conclusions and future work are drawn.

2.The Rule Induction Algorithm

$(LP)^2$ learns from a training corpus where a user has highlighted the information to be extracted with differ-

ent SGML tags. It induces symbolic rules that insert SGML tags into texts in two steps:

1. Sets of **tagging rules** are induced by bottom-up generalization of tag instances found in the training corpus. Shallow knowledge about NLP is used in the generalization process.
2. **Correction rules** are induced that refine the tagging by correcting mistakes and imprecision

This section presents and discusses these two steps.

2.1 Inducing Tagging Rules

A tagging rule is composed of a left hand side, containing a pattern of conditions on a connected sequence of words, and a right hand side that is an action inserting an SGML tag in the texts. Each rule inserts a single SGML tag, e.g. <*speaker*>. This makes $(LP)^2$ different from many adaptive IE algorithms, whose rules recognize whole slot fillers (i.e. insert both <*speaker*> and </*speaker*> [Califf 1998; Freitag 1998]) or even multi slots [Soderland 1999]. The tagging rule induction algorithm uses positive examples from the training corpus for learning rules. Positive examples are the SGML tags inserted by the user. All the rest of the corpus is considered a pool of negative examples. For each positive example the algorithm: (1) builds an initial rule, (2) generalizes the rule and (3) keeps the *k* best generalizations of the initial rule. In particular $(LP)^2$'s main loop starts by selecting a tag in the training corpus and extracting from the text a window of ***w*** words to the left and ***w*** words to the right. Each information stored in the 2****w*** word window is transformed into a condition in the initial rule pattern, e.g. if the third word in the window is "seminar", the condition on the third word in the pattern will be `word="seminar"`. Each initial rule is then generalized. In the generalization process $(LP)^2$ uses generic shallow knowledge about Natural Language as provided by a morphological analyzer, a POS tagger and a user-defined dictionary (or a gazetteer). A *lexical item* (LexIt in the following) summarizes such knowledge for each word (e.g., companies) via: a lemma (company), a lexical category (noun), case information (lowercase) and a list of user defined classes as defined by a user-defined dictionary or a gazetteer (if available). An initial rule and associated information in the LexIts is in Table 1.

Generalization consists in the production of a set of rules derived by relaxing constraints in the initial rule pattern. Conditions are relaxed both by reducing the pattern in length and by substituting constraints on words with constraints on some parts of the additional knowledge. Table 2 shows one of the many generalizations for rule in table 1. The last step of the algorithm is the selection of the best generalizations. Each generalization is tested on the training corpus and an accuracy score *L=wrong/matched* is calculated. For each initial instance the *k* best generalizations are kept that: (1) report better accuracy; (2) cover more positive examples; (3) cover different parts of input[1]; (4) have an error rate that is less than a specified threshold.

word index	**Condition**	**Associated information**				**Action**
	word	lemma	LexCat	case	SemCat	Tag
1	the	the	Art	low		
2	seminar	seminar	Noun	low		
3	at	at	Prep	low		**<stime>**
4	4	4	Digit	low		
5	pm	pm	Other	low	timeid	
6	will	will	Verb	low		

Table 1: Starting rule (with associated NLP knowledge) inserting `<stime>` in the sentence ``the seminar at `<stime>` 4 pm will…".

The other generalizations are discarded. Retained rules become part of the **best rules pool**. When a rule enters the best rules pool, all the instances covered by the rule are removed from the positive examples pool, i.e. covered instances will no longer be used for rule induction ($(LP)^2$ is a covering algorithm). Rule induction continues by selecting new instances and learning rules until the pool of positive examples is void.

Word index	**Condition**					**Action**
	Word	Lemma	LexCat	Case	SemCat	Tag
3		at				**<time>**
4			digit			
5					timeid	

Table 2: A generalization for rule in table 1. The pattern is relaxed in length (conditions on words 1, 2 and 6 were removed) and conditions on the other words were substituted by other constraints.

2.1.1 Learning Contextual Rules

When applied on the test corpus, the best rules pool provides good results in terms of precision, but limited effectiveness in terms of recall. This means that such rules insert few tags (low recall), and that such tags are generally correct (high precision). Intuitively this is because the absolute reliability required for rule selection is strict, thus only some of the induced rules will match it. In order to reach acceptable effectiveness, it is necessary to identify additional rules able to raise recall without affecting precision. $(LP)^2$ recovers some of the rules not selected as best rules and tries to constraint their application to make them reliable. Constraints on rule application are derived by exploiting interdependencies among tags. As mentioned, $(LP)^2$ learns rules for inserting tags (e.g., <*speaker*>) independently from other tags (e.g., </*speaker*>). But tags are not independent. There are two ways in which they can influence each other: (1) tags represent slots, therefore $\langle tag_x \rangle$ always requires $\langle /tag_x \rangle$; (2) slots can be concatenated into linguistic patterns and therefore the presence of a slot can be a good indicator of the presence of another, e.g.

[1] Rules derived from the same seed cover the same portions of input when ineffective constraints are present.

</speaker> can be used as "anchor tag" for inserting <stime>[2]. In general it is possible to use <tag_x> to introduce <tag_y>. (LP)^2 is not able to use such contextual information, as it induces single tag rules. The context is reintroduced in (LP)^2 as an external constraint used to improve the reliability of unreliable rules. In particular, (LP)^2 reconsiders low precision non-best rules for application in the context of tags inserted by the best rules only. For example some rules will be used only to close slots when the best rules were able to open it, but not close it (i.e., when the best rules are able to insert <tag_x> but not </tag_x>). Selected rules are called **contextual rules**. As example consider a rule inserting a </speaker> tag between a capitalized word and a lowercase word. This is not a best rule as it reports high recall/low precision on the corpus, but it is reliable if used only to close an open <speaker>. Thus it will only be applied when the best rules have already recognized an open <speaker>, but not the corresponding </speaker>. Area of application is the part of the text following a <speaker> and within a distance minor or equal to the maximum length allowed for the present slot[3]. "Anchor tags" used as contexts can be found either to the right of the rule space application (as in the case above when the anchor tag is <speaker>), or to the left as in the opposite case (anchor tag is </speaker>). Detailed description of this process can be found in [Ciravegna 2000a]. Reliability for contextual rules is computed by using the same error rate used for best rules, but only matches in controlled contexts are counted.

In conclusion the sets of tagging rules (LP)^2 induces are both the best rule pool and the contextual rules. Figure 3 shows the whole algorithm for tagging rule induction.

```
Loop for instance in initial-instances
 unless already-covered(instance)
 loop for rule in generalise(instance)
  test(rule)
  if best-rule?(rule)
   then insert(rule, bestrules)
        cover(rule, initial-instances)
   else loop for tag in tag-list
     if test-in-context(rule,tag,:right)
      then select-contxtl(rule,tag,:right)
     if test-in-context(rule,tag,:left)
      then select-contxtl(rule,tag,:left)
```

Figure 3: The final algorithm for rule tagging induction.

2.2 Inducing Correction Rules

Tagging rules when applied on the test corpus report some imprecision in slot filler boundary detection. A typical mistake is for example "at <time> 4 </time> pm", where "pm" should have been part of the time expression. For this reason (LP)^2 induces rules for shifting wrongly positioned tags to the correct position. It learns from the mistakes made in applying tagging rules on the training corpus. Shift rules consider tags misplaced within a distance *d* from the correct position. Correction rules are identical to tagging rules, but (1) their patterns match also the tags inserted by the tagging rules and (2) their actions shift misplaced tags rather than adding new ones. An example of an initial correction rule for shifting </stime> in "at <stime> 4 </stime> pm" is shown in table 4.

The induction algorithm used for the best tagging rules is also used for shift rules: initial instance identification, generalization, test and selection. "Wrong Tag" and "Correct Tag" conditions are never relaxed. Positive (correct shifts) and negative (wrong shifts of correctly assigned tags) are counted. Shift rules are accepted only if they report an acceptable error rate.

3. Extracting Information

In the testing phase information is extracted from the test corpus in four steps: initial tagging, contextual tagging, correction and validation. The best rule pool is initially used to tag the texts. Then contextual rules are applied in the context of the introduced tags. They are applied until new tags are inserted, i.e. some contextual rules can match also tags inserted by other contextual rules. Then correction rules correct some imprecision. Finally each tag inserted by the algorithm is validated. There is no meaning in producing a start tag (e.g. <speaker>) without its corresponding closing tag (</speaker>) and vice versa, therefore uncoupled tags are removed in the validation phase.

Condition			Additional Information			
word	Wrong tag	correct tag	lemma	LexCat	case	SemCat
at			at	prep	low	
4	</stime>		4	digit	low	
pm		</stime>	pm	other	low	timeid

Table 4: A a correction rule. The action (not shown) shifts the tag from the wrong to the correct position.

4. Experimental Results

(LP)^2 was tested in a number of tasks in two languages: English and Italian. In each experiment (LP)^2 was trained on a subset of the corpus (some hundreds of texts, depending on the corpus) and the induced rules were tested on unseen texts. Here we report about results on two standard tasks for adaptive IE: the CMU seminar announcements and the Austin job announcements[4]. The first task consists of uniquely identifying speaker name, starting time, ending time and location in 485 seminar announcements [Freitag 1998]. Table 5 shows the overall accuracy obtained by (LP)^2, and compares it with that obtained by other state of the art algorithms. (LP)^2 scores the best results in the task. It definitely outperforms other symbolic approaches (+8.7% wrt Rapier[Califf

[2] In the following we just make examples related to the first case as it is more intuitive.

[3] The training corpus is used for computing the maximum filler length for each slot.

[4] Corpora available at www.isi.edu/muslea/RISE/index.html

1998], +21% wrt to Whisk[Soderland 1999]), but it also outperforms statistical approaches (+2.1% wrt BWI [Freitag and Kushmerick 2000] and +4% wrt HMM [Freitag and McCallum 1999]). Moreover $(LP)^2$ is the only algorithm whose results never go down 75% on any slot (second best is BWI: 67.7%).

	$(LP)^2$	BWI	HMM	SRV	Rapier	Whisk
speaker	**77.6**	67.7	76.6	56.3	53.0	18.3
location	75.0	76.7	**78.6**	72.3	72.7	66.4
stime	99.0	**99.6**	98.5	98.5	93.4	92.6
etime	95.5	93.9	62.1	77.9	**96.2**	86.0
All Slots	**86.0**	**83.9**	**82.0**	**77.1**	**77.3**	**64.9**

Table 5: F-measure (β=1) obtained on CMU seminars. Results for algorithms other than $(LP)^2$ are taken from [Freitag and Kushmerick 2000]. We added the comprehensive ALL SLOTS figure, as it allows better comparison among algorithms. It was computed by:

$$\frac{\Sigma_{slot}\ (\text{F-measure} * \text{number of possible slot fillers})}{\Sigma_{slot}\ \text{number of possible slot fillers}} * 100$$

Concerning $(LP)^2$ results from a 10 cross-folder experiment using half of the corpus for training. F-measure calculated via the MUC scorer [Douthat 1998]. Average training time per run: 56 min on a 450MHz computer. Window size *w*=4.

A second task concerned IE from 300 Job Announcements taken from misc.jobs.offered [Califf 1998]. The task consists of identifying for each announcement: message id, job title, salary offered, company offering the job, recruiter, state, city and country where the job is offered, programming language, platform, application area, required and desired years of experience, required and desired degree, and posting date. The results obtained on such a task are reported in table 6. $(LP)^2$ outperforms both Rapier and Whisk (Whisk obtained lower accuracy than Rapier [Califf 1998]). We cannot compare $(LP)^2$ with BWI as the latter was tested on a very limited subset of slots. In summary, $(LP)^2$ reaches the best results on both the tasks.

Slot	$(LP)^2$	Rapier	BWI	*Slot*	$(LP)^2$	Rapier
id	**100**	97.5	100	platform	**80.5**	72.5
title	**43.9**	40.5	50.1	application	**78.4**	69.3
company	**71.9**	69.5	78.2	area	**66.9**	42.4
salary	62.8	**67.4**		req-years-e	**68.8**	67.1
recruiter	**80.6**	68.4		des-years-e	60.4	**87.5**
state	84.7	**90.2**		req-degree	**84.7**	81.5
city	**93.0**	90.4		des-degree	65.1	**72.2**
country	81.0	**93.2**		post date	**99.5**	**99.5**
language	**91.0**	80.6		**All Slots**	**84.1**	75.1

Table 6: F-measure (β=1) obtained on the Jobs domain using half of the corpus for training.

5. Discussion

$(LP)^2$'s main features that are most likely to contribute to the excellence in the experiments are: (1) the induction of symbolic rules (see the conclusions), (2) rule generalization via shallow NLP, (3) the use of single tag rules and (4) the use of correction.

$(LP)^2$ induces rules by instance generalization. Generalization is also used in SRV, Rapier and Whisk. It allows reducing data sparseness by capturing some general aspects beyond the simple flat word structure. Shallow NLP is the basis for generalization in $(LP)^2$. Morphology allows overcoming data sparseness due to number/gender word realizations, while POS tagging information allows generalization over lexical categories. In principle such type of generalization produces rules of better quality than those matching the flat word sequence, rules that tend to report better effectiveness on unseen cases. This is because both morphology and POS tagging are generic NLP processes performing equally well on unseen cases; therefore rules relying on their results apply successful on unseen cases. This intuition was confirmed experimentally: $(LP)^2$ with generalization ($(LP)^2_G$) definitely outperforms a version without generalization ($(LP)^2_{NG}$) on the test corpus, while having comparable results on the training corpus (+57% on the speaker field, +28% on the location field, +11% overall on the CMU task). Moreover in $(LP)^2_G$ the covering algorithm converges more rapidly than in $(LP)^2_{NG}$, because its rules tend to cover more cases. This means that $(LP)^2_G$ need less examples in order to be trained, i.e., rule generalization also allows reducing the training corpus size. Not surprisingly the role of shallow NLP in the reduction of data sparseness is more relevant on semi-structured or free texts (such as the CMU seminars) than on documents with highly standardized language (e.g. HTML pages, or the job announcement task). During the rule selection phase $(LP)^2$ is able to adopt the right level of NLP information for the task at hand: in an experiment on texts written in mixed Italian/English we used an English POS tagger that was completely unreliable on the Italian part of the input. $(LP)^2_G$ reached the same effectiveness of $(LP)^2_{NG}$, because the rules using the unreliable NLP information were automatically discarded. This shows that the use of NLP is always a plus, never a minus.

The separate recognition of tags is an aspect shared by BWI, while HHM, Rapier and SRV recognized whole slots and Whisk recognizes multislots. Separate tag identification allows further reduction of data sparseness, as it better generalizes over the coupling of slot start/end conditions. For example in order to learn patterns equivalent to the regular expression (`` `at'|`starting from' ``)DIGIT(`` `pm'|`am' ``), $(LP)^2$ just needs two examples, e.g., `` `at'+`pm' `` and `` `starting from'+`am' ``, because the algorithm induces two independent rules for <*time*> (`` `at' + `starting from' ``) and two for </*time*> (`` `am' + `pm' ``). In a slot-oriented rule learning strategy four examples (and four rules) will be needed, i.e. `` `at'+`pm' ``, `` `at'+`am' ``, `` `starting from' +`pm' ``, `` `starting from'+`am' ``. In a multislot approach the problem is worst and the number of training examples needed increases drastically [Ciravegna

2000a].
Another reason for the good experimental results relies in the use of a correction step. Correction is useful in recognizing slots with fillers with high degree of variability (such as the speaker in the CMU experiment), while it does not pay on slots with highly standardized fillers (such as many slots in the Jobs task). $(LP)^2$ using correction rules reports 7% more in terms of accuracy on </speaker> than $(LP)^2$ without correction. Imprecision in tagging was also reported by [Califf 1998] who noted up to 5% imprecision on some slots (but she did not introduce any correction steps in Rapier).

Slot	PRE	REC	F-measure	*Slot*	PRE	REC	F-measure
Name	97	82	88.9	Email	92	71	80.1
Street	96	71	81.6	Tel.	93	75	83.0
City	90	90	90	Fax	100	50	66.6
Prov.	97	92	94.4	Zip	100	90	94.7
Zip	100	90	94.7				

Table 7: Results of a blind test on 50 resumees. This is not a simple named entity recognition task. A resumee may contain many names and addresses (e.g. previous work addresses, name of referees or thesis supervisors and their addresses). The system had to recognize the correct ones.

6. Developing real world applications

$(LP)^2$ was developed as a research prototype, but it quickly turned out to be suitable for real world applications. An industrial system based on $(LP)^2$, *LearningPinocchio*, was developed. Recently *LearningPinocchio* has been used in a number of industrial applications. Moreover licenses have been released to external companies for further application development. This section reports about some industrial applications we have directly developed. The system is used for extracting information from professional resumees written in English. It is used on the results of a spider that surfs the Web to retrieve professional resumees. The spider classifies resumees by topics (e.g. computer science). *LearningPinocchio* extracts the relevant information and its output is used to populate a database. Table 7 shows some results obtained in such task. Application development time for the IE task required about 24 person hours for scenario definition and revision (the scenario was refined by tagging some texts in different ways and discussing among annotators). Further 10 person hours were needed for tagging about 250 texts. The rule induction process took 72 hours on a 450MHz machine, with window size w=4. Finally system results validation required four person hours.

TAG	**F(1)**	**TAG**	**F(1)**
Geograph Area	0.70	Organiz. Name	0.86
Currency	0.85	Company Share	
Stock Exchange		Name	0.85
Name	0.91	Type	0.92
Index	0.97	Category	0.86
ALL SLOTS 0.87			

Table 8: Results of blind test on financial news (300 texts).

Two other applications were developed for Kataweb, a major Italian Internet portal. The goal was to extract information from both financial news and classified ads written in Italian and published on the portal pages. *LearningPinocchio* is used both to generate hyperlinks for cross-referencing texts and to retrieve texts querying the content. The application is currently under final test at the customer's site. Table 8 shows experimental results on financial news.

7. Conclusions and Future Work

$(LP)^2$ is a successful algorithm. On the one hand it outperforms the other state of the art algorithms on two very popular IE tasks. It is important to stress the fact that $(LP)^2$ outperforms also statistical approaches, because in the last years the latter largely outperformed symbolic approaches. There is a clear advantage in using symbolic rules in real world applications. It is possible to inspect the final system results and manually add/modify/remove rules for squeezing additional accuracy (it was not done in the scientific experiments, but it was in the applications).
On the other hand $(LP)^2$ was the basis for building *LearningPinocchio*, a tool for building adaptive IE applications that is having a considerable commercial success. This shows that adaptive IE is able produce tools suitable for building real world applications by a final user by using only analyst's knowledge.
Future work on $(LP)^2$ will involve both the improvement of rule formalism expressiveness and the further use of shallow NLP for generalization. Concerning the improvement in rule formalism expressiveness we plan to include some forms of Kleene-star and optionality operators. Such improvement has shown to be very effective in both BWI and Rapier. Concerning the use of shallow NLP for generalization (i.e., one of the keys of the success in $(LP)^2$) there are two possible improvements. On the one hand $(LP)^2$ will be used in cascade with a Named Entity Recognizer (also implemented by using $(LP)^2$). This will allow further generalization over named entity classes (e.g., the speaker is a person, so it is possible to generalize over such class in the rules). On the other hand $(LP)^2$ is compatible with forms of shallow parsing such as chunking. It is then possible to preprocess the texts with a chunker and to insert tags only at the chunk borders. This is likely to improve precision in border identification.
An interesting question concerns the limits of the tagging-based IE approach used by many adaptive systems, $(LP)^2$ included. Classic MUC-like IE is based on template filling. Template filling is more complex than tagging, as it implies to decide about both coreference of expressions (are "John A. Smith" and "J. Smith" the same person? Two seminars have been identified in a text: are they separate events or are they coreferring?), and slot pairing (two seminars and two speakers have been identified: which is the speaker of the first semi-

nar?). $(LP)^2$ is able to apply default strategies for template merging that solve simple cases of coreferences and slot pairing. Such strategies are powerful enough to cope with many real world tasks, but in some other cases they are not effective enough. For example in the resumees application the customer was interested in retrieving also the triples degree/university/year. *LearningPinocchio* was able to correctly highlight such information, but often it was not able to pair them correctly, therefore they were not used to populate the database, but only to index texts. Even if some ad hoc strategies would have probably solved the problem in the specific case, it is quite clear that this is a major limitation in the approach. Classical MUC-like IE systems use sophisticated strategies for coreference resolution and template merging, very often based on a mix of NLP knowledge and domain knowledge [Humphreys *et al.* 1998]. Two problems prevent the use of such techniques. On the one hand deep NLP is not effective in many applications in the Internet realm (e.g. how can you parse an e-mail?). On the other hand it is not clear how to elicit the domain knowledge for coreference from an analyst (adaptability via analyst's knowledge is a strong constraint as mentioned above). Adaptive template filling is an issue worth exploration that we are currently investigating. We work in the direction of further using shallow NLP for improving template filling and merging. To some extent this is also a step in the direction of bridging the gap between classical NLP based IE systems and fully adaptive systems.

Acknowledgments

I developed $(LP)^2$ and *LearningPinocchio* at ITC-Irst, Centro per la Ricerca Scientifica e Tecnologica, Trento, Italy. LearningPinocchio is property of ITC-Irst, see http://ecate.itc.it:1025/cirave/LEARNING/home.html.
The financial application mentioned above was jointly developed with Alberto Lavelli. Thanks to Daniela Petrelli for revising this paper. Errors, if any, are mine.

References

[Califf 1998] Mary E. Califf, Relational Learning Techniques for Natural Language IE, *Ph.D. thesis*, Univ. Texas, Austin, www.cs.utexas.edu/users/mecaliff

[Cardie 1997] Claire Cardie, `Empirical methods in information extraction', *AI Journal*, 18(4), 65-79, 1997.

[Ciravegna *et al.* 2000] Fabio Ciravegna, Alberto Lavelli, and Giorgio Satta, `Bringing information extraction out of the labs: the Pinocchio Environment', in *ECAI2000, Proc. of the 14th European Conference on Artificial Intelligence*, ed., W. Horn, Amsterdam, 2000. IOS Press.

[Ciravegna 2000a] Fabio Ciravegna, `Learning to Tag for Information Extraction from Text' in F. Ciravegna, R. Basili, R. Gaizauskas (eds.) *ECAI Workshop on Machine Learning for Information Extraction*, Berlin, August 2000. (www.dcs.shef.ac.uk/~fabio/ecai-workshop.html)

[Douthat 1998] Aaron Douthat, `The message understanding conference scoring software user's manual', in *the 7th Message Understanding Conf.*, www.muc.saic.com

[Freitag 1998] Dayne Freitag, `Information Extraction from HTML: Application of a general learning approach', *Proc. of the 15th National Conference on Artificial Intelligence (AAAI-98)*, 1998.

[Freitag and McCallum 1999] Dayne Freitag and Andrew McCallum: `Information Extraction with HMMs and Shrinkage', *AAAI-99 Workshop on Machine Learning for Information Extraction*, Orlando, FL, 1999, www.isi.edu/~muslea/RISE/ML4IE/

[Freitag and Kushmerick 2000] Dayne Freitag and Nicholas Kushmerick, `Boosted wrapper induction', in F. Ciravegna, R. Basili, R. Gaizauskas (eds.) *ECAI2000 Workshop on Machine Learning for Information Extraction*, Berlin, 2000, (www.dcs.shef.ac.uk/~fabio/ecai-workshop.html)

[Grishman 1997] Ralph Grishman, ` Information Extraction: Techniques and Challenges, . In *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology*, in M.T. Pazienza, (ed.), Springer, 97.

[Humphreys *et al.* 1998] K. Humphreys, R. Gaizauskas, S. Azzam, C. Huyck, B. Mitchell, H. Cunningham, Y. Wilks: `Description of the University of Sheffield LaSIE-II System as used for MUC-7'. In *Proc. of the 7th Message Understanding Conference*, 1998 (www.muc.saic.com).

[Kushmerick *et al.* 1997] N. Kushmerick, D. Weld, and R. Doorenbos, `Wrapper induction for information extraction', *Proc. of 15th International Conference on Artificial Intelligence, IJCAI-97*, 1997.

[Miller *et al.* 1998] S. Miller, M. Crystal, H. Fox, L. Ramshaw, R. Schwartz, R. Stone and R. Weischedel, `BBN: Description of the SIFT system as used for MUC-7', In *Proc. of the 7th Message Understanding Conference*, 1998 (www.muc.saic.com).

[Muslea *et al.* 1998] I. Muslea, S. Minton, and C. Knoblock, `Wrapper induction for semi-structured, web-based information sources', in *Proc. of the Conference on Autonomous Learning and Discovery CONALD-98*, 1998.

[Soderland 1999] Steven Soderland, `Learning information extraction rules for semi-structured and free text', *Machine Learning*, (1), 1-44, 1999.

[Yangarber *et al.* 2000] Roman Yangarber, Ralph Grishman, Pasi Tapanainen and Silja Huttunen: ``Automatic Acquisition of Domain Knowledge for Information Extraction'' In *Proc. of COLING 2000, 18th Intern. Conference on Computational Linguistics*, Saarbrücken, 2000.