$(LP)^2$: Rule Induction for Information Extraction Using Linguistic Constraints

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

Technical Report CS-03-07
September 29, 2003

Abstract

Machine learning has been widely used in information extraction from texts in the last years. Two directions of research can be identified: wrapper induction (WI) and NLP-based methodologies. WI techniques have historically made scarce use of linguistic information and their application is mainly limited to rigidly structured documents. NLP-based methodologies tend to be brittle when linguistic information is unreliable and their application to structured web documents has been largely unsuccessful. In this paper we describe $(LP)^2$, an adaptive algorithm able to cope with a number of document types, from free texts to highly structured web pages. Its main feature is the ability to use both document formatting and linguistic information in the learning phase. In a number of experiments on publicly available corpora, $(LP)^2$ reports excellent results with respect to the state of the art. Also the algorithm has been implemented in two successful systems which have been used extensively for both research and application of IE from Web documents. In this paper we describe the algorithm in details focusing on the role of linguistic information in the rule induction process. We also quantify the contribution of linguistic information to the effectiveness and robustness of the analysis, especially in cases where it can be unreliable.

Introduction

The classic tasks of Information Extraction from text (IE) include Named Entity Recognition (NER) and event extraction (template filling). The former requires identifying and classifying objects belonging to a limited amount of very high-level classes (person, organization, location, etc.), while the latter requires capturing more or less complex event structures. The current state of the art includes algorithms using a very limited amount of linguistic features for the

former (e.g. via use of statistical methods [MCF+98] or manually written Finite State Grammars [AHB+93]). Systems performing complex event extraction tend to use complex linguistic information manually encoded in system grammars [HGA+98a] [AHB+93]. A different kind of task has been defined by the wrapper induction community, requiring the identification of implicit events and relations. For example Freitag [Fre98b] defines the task of extracting speaker, start-time, end-time and location from a set of seminar announcement. No explicit mention of the event (the seminar) is done in the annotation. Implicit event extraction is simpler than full event extraction, but has important applications whenever either there is just one event per texts or it is easy to extract strategies for recognizing the event structure from the document [CL03]. The wrapper induction community has also changed the focus on the type of texts to be coped with, moving from the free text (i.e. newspaper articles) only scenarios used in the MUC conferences (e.g. [MUC98]), to ones based on Web documents[MMK98]. Web documents can range from free texts (technical reports, newspaper-like texts) to (semi)structured marked up texts (e.g. partially marked-up or highly structured XML/HTML documents) and even a mixture of them. Linguistically based methodologies used for free texts (e.g. [HGA+98b] [Gri97]) can be difficult to apply or even ineffective on highly structured texts, such as the Web pages produced by databases. They can't cope with the variety of extra linguistic structures such as HTML tags, document formatting, and stereotypical language that convey information in such documents. Wrapper induction algorithms (e.g. [KWD97], [MMK98]) induce simple regular expressions testing the HTML structure of the document when highly structured regular pages are analyzed [KWD97, MMK98], moving to more sophisticated learning strategies when freer documents are analysed [FM99, Fre98b, FK00]. They tend to use pattern matching on strings and structure, without relying on linguistic information for generalisation. For example BWI [FK00] uses capitalization information, a wildcard and a gazetteer to generalize. More linguistically oriented methodologies (including POS tagging, a parser and the use of Wordnet) were attempted in the past in WHISK [Sod99] and RAPIER [CM97], but these algorithms tended to be outperformed by shallower methods. Also Califf showed that the use of Wordnet did not bring any clear advantage to the algorithm.

In this paper we describe $(LP)^2$, an algorithm for wrapper induction that performs implicit event extraction using linguistic analysis. Linguistic information is used for generalizing over the flat word sequence, but the algorithm is also able to exploit the structure of the document (e.g. HTML tags) according to the task at hand. We will show that the use of linguistic information is not disruptive (and in many case pays off) even when unreliable. $(LP)^2$ reports some excellent results on at least three publicly available corpora and was used in two systems, LearningPinocchio[CL03] and Amilcare [Cir03]. The former is a commercial system used in a number of real world applications. The latter is distributed for free for research purposes and was both adopted as support for semi-automatic annotations by two popular annotation tools for the Semantic Web and is currently used by a number of researchers as a baseline for further experiments.

In the following we will:

- describe the algorithm for rule induction (Sections 1 and 2);
- describe experiments on publicly available corpora (Section 3);
- discuss the role of linguistic generalisation (Section 4);
- draw some conclusion and outline some future work (Section 5).

1 The $(LP)^2$ Algorithm

$(LP)^2$ is an algorithm able to perform implicit event recognition on documents of different types, including free, structured and mixed ones. It has its roots in the wrapper induction technology. It induce rules using linguistic information to generalize over the flat word structure, attempting to use the maximum amount of linguistic information that is reliable for the task at hand, mixing them with matches on both strings and HTML structure, if necessary. In other words, the level of linguistic analysis is one of the parameters that the learner tunes for each task. The linguistic information is provided by generic modules and resources that are defined once and are not to be modified to specific application needs by users (which are imagined naive from this point of view). Pragmatically, the measure of reliability used by the learner to tune the level of linguistic analysis is not linguistic correctness (immeasurable by incompetent users) but effectiveness in extracting information using linguistic methods as opposed to using shallower approaches. Unlike previous approaches in which different algorithm versions with different linguistic competence were tested in parallel and the most effective version is chosen [Sod99], $(LP)^2$ learns which is the best strategy for each information or context separately. For example, it might decide that parsing is the best strategy for recognizing the speaker in a specific application on seminar announcements but not the best strategy to spot the seminar location or starting time. This is quite useful on mixed type documents such as the one in figure 1 where structured and unstructured lists are coexisting with free text.

Rules are learnt by generalising over a set of examples marked via XML tags identifying the information to be extracted by the system in a training corpus. $(LP)^2$ requires a linguistic preprocessor that analyses documents using generic linguistic modules, for example performing tokenization, morphological analysis, part of speech tagging, gazetteer lookup, generic dictionary lookup, Named Entity Recognition, parsing, ontological reasoning, etc. During rule induction, $(LP)^2$ induces rules able to replicate the XML annotation in the corpus. Input in training mode is a preprocessed annotated training corpus. Output is a set of rules.

The algorithm induces symbolic rules that insert XML tags into texts in two steps (see Figure 2):

- Sets of **annotation rules** are induced that insert a preliminary annotation.

Figure 1: An example of a document containing a mixture of text types: free text is located in the center, while structured HTML lists can be seen on the left hand side and a semi structured list on the center right (a list of titles which are mainly fragmented sentences). A flexible methodology for IE is needed in order to extract information from the different parts.

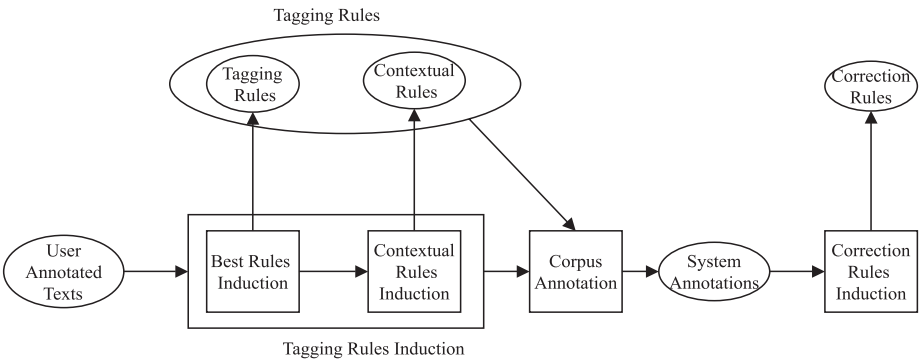

Figure 2: Rule induction steps.

- **Correction rules** are induced that refine the annotation by correcting mistakes and imprecision.

1.1 Inducing Annotation Rules

An annotation rule is composed of a left hand side, containing a pattern of conditions on a sequence of adjacent words, and a right hand side that is an action inserting an XML tag in the texts. Each rule inserts a single XML 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>` [CM97, Fre98a]) or even multiple slots [Sod99, FM99]. The induction algorithm uses positive examples from the training corpus for learning rules. Positive examples are the XML 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 empty rule, (2) progressively adds constraints to the rule (3) keeps the k best rules. 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 piece of information stored in the 2*w word window is transformed into a condition, e.g. if the third word in the window is "seminar", the condition is added to the pool of possible conditions that `Word`$_3$`="seminar"`. Conditions are not limited to word matching, but include linguistic information associated with each word. For example, the condition `Word`$_3$`.POS="noun"` checks if the third word is a noun. Such information is provided by the linguistic preprocessor. Each induced rule is tested on the training corpus and an `error rate` *L=wrong/matched* is calculated. generalisations are accepted when they fall into one of the two groups described in the rest of the section: best rules and contextual rules.

Word Index	Conditions						Action
	Word	Lemma	LexCat	Case	Gazetteer	Html Tags	Insert Tag
3		at					<stime>
4			Digit				
5					TimeId		

Figure 3: A potential rule generated for inserting `<stime>` in the sentence "the seminar at `<stime>` 4 pm will..." where the tag to be inserted is "<stime>".

1.1.1 Best Rules

Best rules are meant to be highly reliable annotation rules. A generalisation becomes part of the set of best rules if:

1. it covers at least a minimum number of cases on the training corpus;
2. its error rate is less than a user-defined threshold.

The best rule list is kept constantly sorted so that better rules come first. Rules are sorted by decreasing number of matches (those with more matches come first). Rules with identical number of matches are sorted by increasing error rate (more precise rules come first). Rules with equal number of matches and equal error rate are sorted using the following criterion: if they report a number of matches below a user-defined threshold, then the one with less generic conditions is preferred (e.g. with conditions on words), otherwise the other one is preferred. This heuristics has been selected in order to favour rules supported by more evidence. Data sparseness, a well-known phenomenon in Machine Learning for NLP, can produce rules with limited number of matches on the training corpus. When they are applied on the test corpus, they can produce unreliable results. Rules with more matches have more evidence supporting them and tend to be more reliable at testing time. This is also why we prefer rules with less generic conditions when they report a limited number of matches on the training corpus: matches on words tend to be more predictable than matches on other parts of the information associated to a word. For example a rule matching the sequence $\texttt{Word}_1$=``in'', $\texttt{Word}_2$=``the'', $\texttt{Word}_3$=``afternoon'' tends to over-recognize less than $\texttt{Case}_1$=low, $\texttt{LexCat}_2$=Art, $\texttt{Word}_3$=``afternoon''.

Rules fully subsumed by other better rules (i.e. their matches are also matched by another better rule) are removed from the list[1].

A maximum of k best rules is constantly kept for each instance. This means that when the list reaches size k, every new generalisation is ignored unless it is

[1] Generalisations derived from the same seed rule cover the same portions of input when ineffective or equivalent constraints are present.

able to enter the list in a position $p \leq k$. If it does, the rule previously in position k is removed from the list. The algorithm is summarized in Figure 4.

Rules retained at the end of the generalisation process of one specific seed rule become part of the global *best rules pool*. When a rule enters such pool, all the instances covered by this 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 empty.

1.1.2 Contextual Rules

When applied to 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 a few of the induced rules will match it. In order to reach acceptable effectiveness, it is necessary to identify additional rules able to increase recall without affecting precision. $(LP)^2$ recovers some of the rules not selected as best rules and tries to constrain their application to make them more 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 `<tag`$_x$`>` always requires `</tag`$_x$`>`; (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. `</speaker>` can be used as "anchor tag" for inserting `<stime>`[2]. In general it is possible to use `<tag`$_x$`>` to introduce `<tag`$_y$`>`. $(LP)^2$ as described so far is not able to use such contextual information, as it induces single tag rules. The context is reintroduced as an external constraint used to improve the reliability of inaccurate rules. In particular, low precision non-best rules are reconsidered for application in the context of tags inserted by the best rules. For example some rules will be used only to close slots when the best rules were able to open it, but not to close it (e.g., when the best rules are able to insert `<tag`$_x$`>` but not `</tag`$_x$`>`). Such rules are called contextual rules. As an 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 tag `<speaker>`. Thus it will only be applied when the best rules have already recognized an open tag `<speaker>`, but not the corresponding `</speaker>`. The area of application is the part of the text following a `<speaker>` and within a distance less than 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

[2] In the following we just make examples related to the first case as it is more intuitive, but the algorithm effectively uses also the other case.

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

```
method SelectRule(rule, currentBestPool)
  if (rule.matches≤MinimumMatchesThreshold)
     then return currentBestPool // i.e.  reject(rule)
  if (rule.errorRate≥ErrorRateThreshold)
     then return currentBestPool // i.e.  reject(rule)
  insert (rule, currentBestPool)
  sort(currentBestPool)
  removeSubsumedRules(currentBestPool)
  cutRuleListToSize(currentBestPool, k)
  return currentBestPool

method sort(ruleList)
  sort by decreasing number of matches
  if two rules have equal number of matches
     then sort by increasing error rate
  if two rules have same error rate and number of matches:
     then if one rule has more matches than a threshold
               then prefer the one with more generic conditions
          else prefer the other one
  return ruleList

method removeSubsumedRules(ruleList)
  loop for index1 from 0 to ruleList.size-1
        rule1=ruleList(index1)
        loop for index2 from index1+1 to ruleList.size
              rule2=ruleList(index2)
              if (subsumes(rule1, rule2))
                 then remove (rule2, ruleList)
  return ruleList

method subsumes(rule1, rule2)
  return (rule2.matches is a subset of rule1.matches)

method cutRuleListToSize(list, size)
  return subseq(list, 0, size)
```

Figure 4: The algorithm for deciding if a generalisation can be considered as a best rule.

above when the anchor tag is <speaker>), or to the left as in the opposite case (anchor tag is </speaker>). Acceptability for contextual rules is computed by using the same algorithm used for selecting best rules, but only matches in constrained contexts are counted.

In conclusion the sets of annotation rules $(LP)^2$ induces are both the best rules pool and the contextual rules. Figure 5 shows the whole algorithm for annotation rule induction.

1.2 Inducing Correction Rules

Annotation rules when applied on the training corpus report some imprecision in slot filler boundary detection. A typical mistake is for example "at <time> 4 </time> pm", where "pm" should be 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 on the training corpus. Shift rules consider tags misplaced within a distance d from the correct position. Correction rules are similar to annotation rules, but (1) their patterns match also the tags inserted by annotation rules and (2) their actions shift misplaced tags rather than adding new ones. An example of a correction rule for shifting </stime> in "at <stime> 4 </stime> pm" is shown in Figure 6. The induction and selection algorithm used for the best annotation rules is also used for shift rules: initial condition identification, specialization, test and selection of best k generalisations.

1.3 Extracting Information

In the testing phase information is extracted from the test corpus in four steps (Figure 7): initial annotation, contextual annotation, correction and validation. The best rules pool is initially used to tag the texts. Then contextual rules are applied in the context of the introduced tags. They are iteratively applied until no new tags are inserted, i.e. some contextual rules can match also tags inserted by other contextual rules. Then correction rules adjust some misplaced tags. Finally each tag inserted by the algorithm is validated. It is not meaningful to produce an open tag (e.g. <speaker>) without its corresponding closing tag (</speaker>) and vice versa, therefore uncoupled tags are removed in the validation phase.

2 Induction as Specialization and Pruning

The types of rule mentioned above are all induced by the same algorithm. As mentioned, induced rule patterns contain conditions on word in a window w around each instance. Conditions are either on strings or on information provided by the linguistic preprocessor (e.g. POS tags, Capitalization information, Named Entities, syntactic chunks, etc.). This means that while many similar successful algorithms rely on non-linguistic generalisation (e.g. BWI [FK00]

```
method InduceRules()
  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)

method generalise(instance)
  currentRules={one empty rule}
  newRules={}
  loop for lexitem in instance.pattern
        generalisations=generalize(lexitem)
        loop  for currentCondition in generalisations
              loop for rule in currentRules
                    newRule=copyRule(rule)
                    addCondition(newRule, currentCondition)
                    newRules.add(newRule)
        currentRules=newRules
        newRules={}
  return currentRules

method generalise(lexitem)
  collect newCondition(lexitem.word)
  if (lexitem.lemma!=null)
     then collect newCondition(lexitem.lemma)
           if (lexitem.case!=null))
           then collect newCondition(lexitem.lemma, lexitem.case)
  if (lexitem.case!=null)
     then collect newCondition(lexitem.word, lexitem.case)
           collect newCondition(lexitem.case)
  if (lexitem.lexCat!=null)
     then collect newCondition(lexitem.lexCat)
  if (lexitem.semCat!=null)
     then collect newCondition(lexitem.semCat)
  return collected conditions
```

Figure 5: The final version of the algorithm for rule annotation induction. The `generalise` function produces all the possible generalisations for a rule.

Word Index	Conditions							Action
	Word	Lemma	LexCat	Case	Gazetteer	Html Tags	Wrong Tag	Move Tag to
3		at						
4			Digit				</stime>	
5					TimeId			</stime>

Figure 6: A correction rule for shifting a misplaced tag in the sentence "the seminar at **<stime>** 4 pm will...". The action shifts the tag from the wrong to the correct position, i.e. from before "pm" to after "pm".

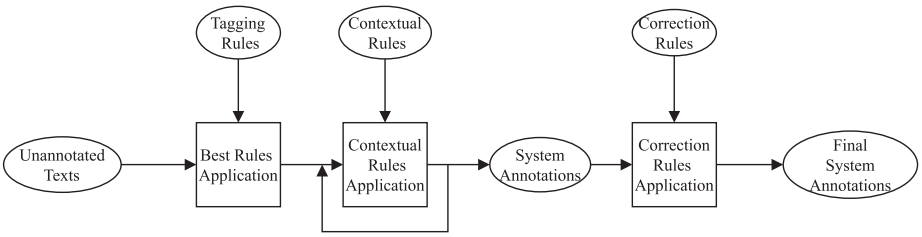

Figure 7: The algorithm for extracting information.

```
InduceRuleFromInstance(instance)
 RuleSet=startRulesFromEmptyRule(instance);
 Loop while notEmpty(RuleSet) {
  Loop for  rule1 in butLast(RuleSet){
       for rule2 in butFirst(RuleSet)
   ruleSet=combineRules(rule1, rule2);
   add(ruleSet, finalRu  leSet);
   }  }
return finalRuleSet;
}

startRulesFromEmptyRule (instance){
 tag=instance.tag
 loop for distance from 0 to 2*  w {
  do word=instance.pattern[distance]
     /*  w-position is the distance between
         the current word and the tag to be
         inserte d */
  collect generateRule(word,tag,  w-position)
 } }
```

Figure 8: The simplified algorithm for non NLP-based rule induction

uses just capitalization, a gazetteer and a wild card), $(LP)^2$ uses linguistic information for generalisation over the flat word level. Linguistic generalisation is important in analysing natural language input, because of data sparseness due to the high flexibility of natural language forms. Avoiding generalisation actually means producing rules covering a limited number of cases each. Such rule set is very likely to produce excellent results on the training corpus, but obtain very limited effectiveness on a test corpus. This is the well known problem of overfitting the training corpus: on the one hand the system learns a number of rules for covering unrelated cases: if such cases are not found as they are, the rule will not apply at testing time (leading to low recall). On the other hand the rule set is sensible to errors in the training data: such errors can either prevent some good rules from being learnt, as they report errors (low recall again during test), or can favour acceptance of bad rules producing spurious results at test time (low precision). It is therefore important to:

- generalise over the plain word surface of the training example in order to produce rules able to overcome data sparseness;
- prune the resulting rule set in order to reduce overfitting and - possibly - reducing the search space.

```
InduceRuleFromInstance(instance)
 RuleSet=startRulesFromEmptyRule(instance);
 Loop while notEmpty(RuleSet) {
  NewRuleSetS={}
  Loop for  ruleSet1 in butLast(RuleSet){
       for ruleSet2 in butFirst(RuleSet)
   ruleSet=combineRuleSet(ruleSet1, ruleSet2);
   pruneRuleSet(ruleSet);
   add(ruleSet, newRuleSetS);
   add(ruleSet, finalRuleSets);
   }
  RuleSet=NewRuleSetS;
 }
 PruneRuleSets(finalRuleSets)
}

startRulesFromEmptyRule (instance)
  tag=instance.tag
  Loop for distance from 0 to 2*w {
    word=instance.pattern[distance]
       /* w-position is the distance between
       the current word and the tag to be
       inserted */
    generateRuleSet (word, tag, w-position)
 }
```

```
generateRuleSet(word, tag, position){
 Loop for condit in NLPGeneralisations(word)
  Collect generateRule(condit, tag, position)
}

PruneRuleSet (ruleSet){
  For rule in ruleSet
   if (not(subsumedRule(rule, RuleSet)))
   collect rule
}

subsumedRule (rule, ruleSet){
  loop for ru in ruleSet
   if (ru!=rule)
    if((ru.score<=rule.score) and
       (includes(ru.coverage,rule.coverage)))
        return true;
 return false;
}
```

Figure 9: The complete algorithm for NLP-based rule induction

2.1 Rule Specialization

We have implemented different strategies for rule generalisation. The naive version of the algorithm [Cir01a] generates all the possible rules in parallel. Each generalisation is then tested separately on the training corpus and an error score E=wrong/matched is calculated. For each initial rule, the k best generalisations are kept that: (1) report better accuracy; (2) cover more positive examples; (3) cover different parts of input. The naive generalisation is quite expensive in computational terms. Here we describe a more efficient version of the algorithm that uses a general to specific beam search for the best k rules in a way similar to AQ [MMJL86]. It starts by modelling a specific instance with the most general rule (the empty rule matching every instance) and specialises it by greedily adding constraints. Constraints are added by incrementing the length of the rule pattern, i.e. by adding conditions on terms, starting from the ones nearest to the tag. Figure 10 shows the search space for an example of this type of generalisation for the case in which conditions are set only on words[4]. The algorithm is described in Figure 8. The actual algorithm is more complex than shown, as it also uses linguistic information (see Figure 9). This algorithm is as effective as the naive one, but it allows a very efficient rule testing. As a matter of fact the matches of a specialised rule can be computed as the intersection of

[4]Note that we always start adding conditions to elements near the tag and proceed outwards. One condition is always a wild card that means no conditions on the term. Computationally, this is equivalent to selecting each time the best condition to be added over the broad spectrum of the window (as for example in CN2), but the implementation for testing the generated rules is more efficient because it uses intersections on integers, as shown in figure 10.

the matches of two more general rules, making testing an order of magnitude more efficient.

2.2 Rule Set Pruning

Pruning is performed both for efficiency reasons (so to reduce the search space during generalisation) and to reduce overfitting. Pruning removes rules that either are unreliable, or whose coverage fully overlaps with those of other rules. There are two types of unreliable rules: those performing poorly on the training corpus (i.e. with a high error rate) and those reporting a very limited number of matches on the training corpus, so that it is not easy to foresee what their behaviour at test time. In order to prune the final rule set from the first type of rules, accuracy is tested against a maximum error threshold set by the user before running the algorithm. Rules that do not pass the error rate test are discarded as soon as they are generated and tested. They are no longer considered for IE, but they are considered for further specialisation, as they could become reliable after the addition of more constraints. At the end of training the algorithm tries to optimise the error threshold by restricting the rule selection condition, therefore excluding some other rules. It tests the results of the reduced rule set on the training corpus and stops pruning the moment in which there is a reduction of accuracy (seen as the mean of precision and recall).

Rules are also pruned at induction time when they cover too few cases in the training corpus and therefore they cannot be safely used at test time because the amount of training data does not guarantees a safe behaviour at run time. Such rules are detected at generation time and any further specialisation depending from them is stopped. In this way the search space is reduced and the algorithm is more efficient. For example with a threshold on minimum coverage set to 3 cases, rules 9 and 10 are not generated in the example in Figure 10. Again the user can set a priori the minimum coverage threshold and such threshold will be optimised at the end of training in a way similar to the error threshold optimisation.

There is a further level of pruning based on overlapping rule coverage: all the rules whose coverage is subsumed by that of others more general ones are removed from the selected rule set. The goal is to determine the minimum subset of rules that maximises the accuracy of IE on the training corpus. Rules whose coverage is subsumed by those of other rules can be safely removed, as their contribution to the final results is irrelevant. Such type of pruning requires comparing not only coverage, but also the reliability of the involved rules (otherwise the algorithm would only produce one rule, i.e. the empty initial rule). In general rules subsumed by other rules with the same (or minor) error rate can be safely removed during rule induction. Rules subsumed by ones with worse error rate are pruned when the final error and covering threshold have been determined, as it is necessary to see if the subsuming rule will survive the final pruning process. A risk of overgeneralisation arises in pruning at every level: when two rules cover the same set of examples with the same error rate, choosing the one with the most generic condition when there is limited evidence

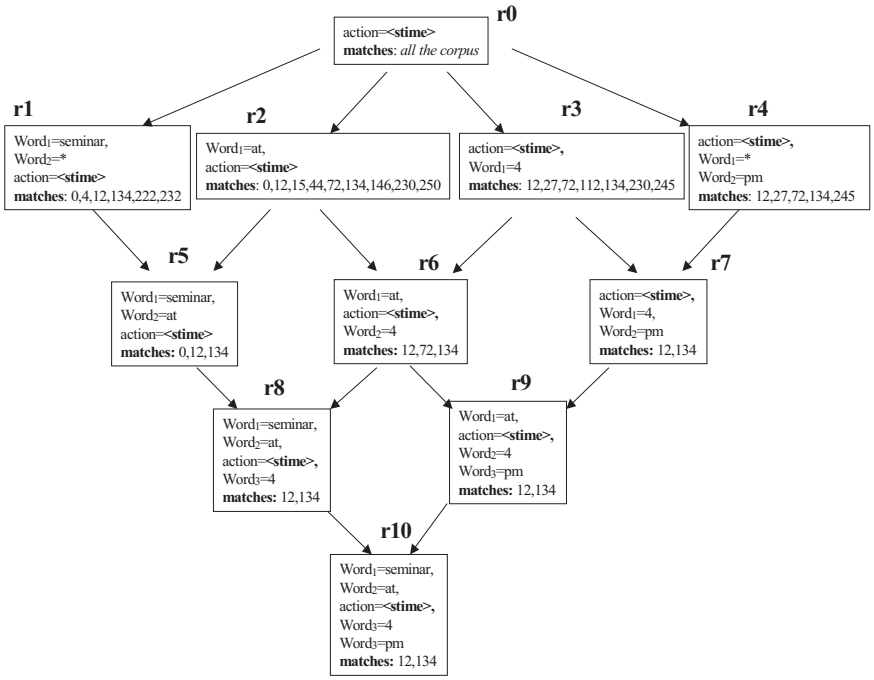

Figure 10: Pattern specialization for "the seminar at <stime> 4 pm will" with window w=2 (conditions on words only). The rule action is inserted in the part of the pattern where the tag is inserted. For example r6 matches "at" + "4" and inserts a tag after "at". The "matches" fields shows positions in the corpus where a rule applies.

in the training corpus risks to produce unreliable rules at test time. In this case the following heuristic is used. If the number of covered cases is limited, then the one with most specific conditions is chosen (e.g. one using condition on words). This is because the training corpus does not provide enough evidence that the rule is reliable and a rule requiring a sequence of words is less likely to produce spurious results at test time than one requiring sequences of conditions on the additional knowledge (e.g. on lexical categories). Otherwise, the rule with the most generic conditions is selected (e.g. testing lexical categories), as it is more likely to provide coverage on the test corpus.

3 Experimental Results

$(LP)^2$ was tested in a number of tasks in two languages: English and Italian. In each experiment the algorithm 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. In this section we report about results on three standard tasks for

	Speaker	Location	Start Time	End Time	**All Slots**
$(LP)^2$	77.6	75.0	99.0	95.5	**86.0**
SNoW	73.8	75.2	99.6	96.3	**85.3**
Max Entropy	65.3	82.3	99.6	94.5	**85.0**
BWI	67.7	76.7	99.6	93.9	**83.9**
HMM	76.6	78.6	98.5	62.1	**82.0**
SRV	56.3	72.3	98.5	77.9	**77.1**
Rapier	53.0	72.7	93.4	96.2	**77.3**
Whisk	18.3	66.4	92.6	86.0	**64.9**

Table 1: F-measure (beta =1) obtained on CMU seminars. For $(LP)^2$ the experiments are obtained as the average results of 10 sessions of experiments using a random half of the corpus for training and the rest of the corpus for testing. We did not use any named entity recognition in the experiments. With NER, Speaker grows to 80.9% and Location to 79.8%.

Table 2: F-measure (b=1) for misc.jobs.offered using 1/2 corpus for training.

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

adaptive IE: the CMU seminar announcements, the Austin job announcements and the computer course testbed.

The first task consists of uniquely identifying speaker name, starting time, ending time and location in 485 seminar announcements [Fre98b]. Table 3 shows the overall accuracy obtained, and compares it with that obtained by other state of the art algorithms. $(LP)^2$ scores the best overall results in the task. It definitely outperforms other symbolic NLP-based approaches (+8.7% with respect to Rapier[CM97], +21% w.r.t. to Whisk[Sod99]). Moreover $(LP)^2$ is the only algorithm whose results never go down 75% on any slot (second best is SNOW: 73.8%). Please note that most of the papers did not report the comprehensive ALL SLOTS figure, so we added it as it allows better comparison among algorithms. It was computed by:
$\Sigma_{slot}\ (F - measure\ *\ number\ of\ possible\ slot\ fillers) * 100/$
$\Sigma_{slot}\ number\ of\ possible\ slot\ fillers.$
For $(LP)^2$, the results were obtained in ten experiments using a random half of the corpus for training and the rest for testing. For preprocessing we just

	Precision	Recall	F-measure
HMM	60.85	62.06	61.45
$(LP)^2$	69.74	62.12	65.71
STALKER	19.77	49.55	28.26
SRV	74.37	59.74	66.08

Table 3: Experimental results on the Computer Science Course corpus. The results show the overall accuracy reached by the different systems using 50% of the corpus for training.

used capitalization information and pos tagging. No other algorithm used a Named Entity Recognizer, so we avoided using it. We will come back to the importance of the NER later in the paper. F-measure calculated via the MUC scorer [Dou98]. Average training time per run: 5.2 minutes per run on a 2.1GHz PC. Window size w=5. The learning curve is partially shown in figure 11. Note that the task above (as well as those below) requires implicit relation recognition, not simply Named Entity Recognition, as information is classified using the role they play in an implicit event, not in isolation. For example the speaker of a seminar is not the only person name mentioned in the message (e.g. the sender is another one, the secretary organizing the booking may be another one, etc.), the same applies to the time expressions (stime and etime are not the only time expressions in the messages. etc.).

A second task concerned IE from 300 Job Announcements [CM97] taken from `misc.jobs.offered`. 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 3. $(LP)^2$ outperforms both Rapier and Whisk (Whisk obtained lower accuracy than Rapier [CM97]). We cannot compare $(LP)^2$ with BWI as the latter was tested on a very limited subset of slots.

The third experiment was performed by a group from NCRS "Demokritos" in Greece [SPSS03] using Amilcare [Cir03]. It consists in extracting information from 101 Web pages describing Computer Science courses collected from different university sites in the context of the WebKb project. The information to be extracted was course number, title and instructor. Results for four systems are reported in Table 3.

In summary, $(LP)^2$ reaches excellent results in all tasks.

4 Discussion

$(LP)^2$'s main features that are most likely to contribute to the excellence in the experiments are: (1) the use of single tag rules, (2) the use of a correction phase, and (3) rule generalisation via shallow NLP processing. The separate recognition of tags is an aspect shared by BWI, while HMM, Rapier and SRV recognized

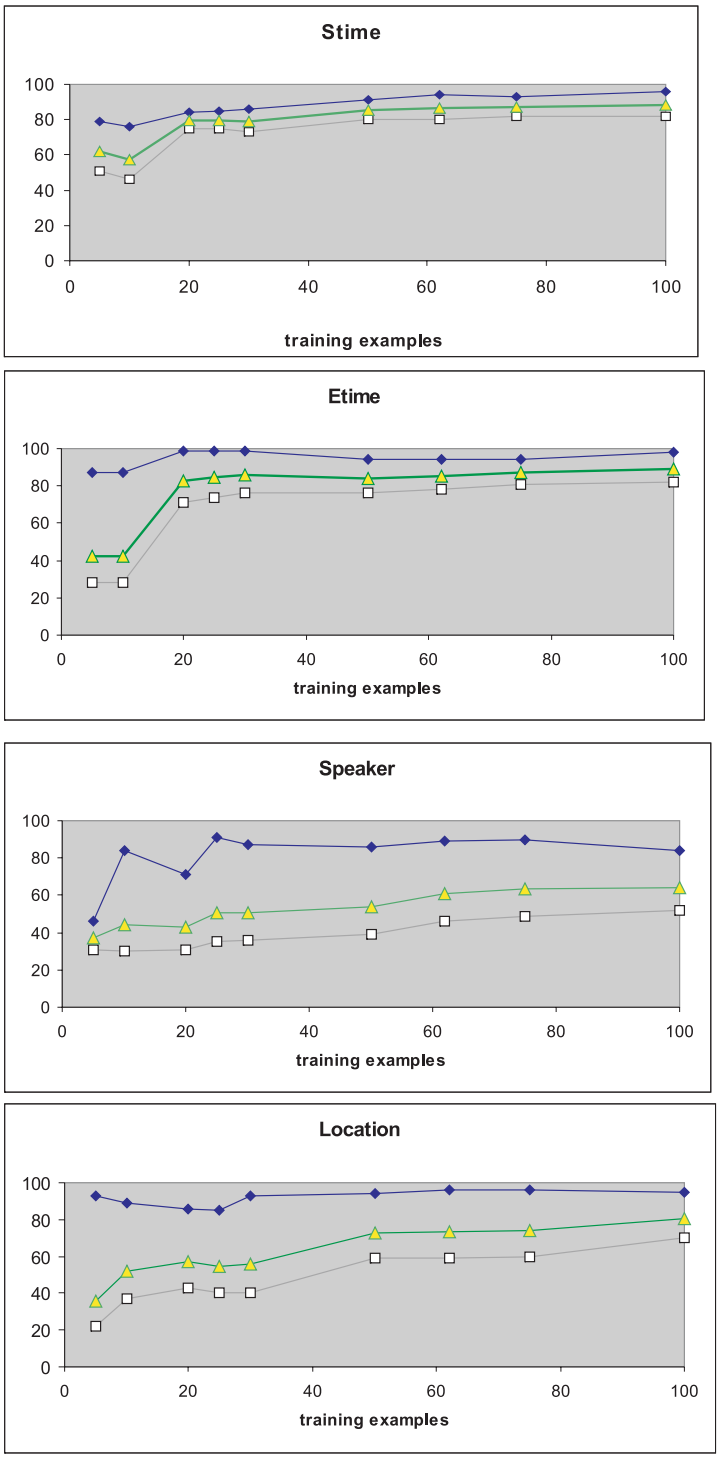

Figure 11: The learning curves for the CMU seminar announcement showing precision (topmost curve in all graphs), recall (bottom most curve) and f-measure (middle curve)

whole slots and Whisk recognizes multi-slots. Separate tag identification allows 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 4 pm'/'starting from 9 am'
```

because the algorithm induces two independent rules for <stime> ('at' and 'starting from') and two for </stime> ('am' and 'pm'). A slot-oriented rule learning strategy needs four examples:

```
at 4 pm
starting from 4 pm
at 9 am
starting from 9 am.
```

In a multi-slot approach the problem is worst and the number of training examples needed increases drastically: it is necessary that all the four examples mentioned above are found in all the multi-slot contexts. This is probably the explanation for the very low recall obtained by Whisk in the CMU seminar task (about 45% less than $(LP)^2$'s on the "speaker" slot). Single tag rules are likely to provide higher recall, with a potential reduction in precision due to some overgeneration (never registered in the experiments, though).

$(LP)^2$ is the only system that adopts a correction step in order to remove some imprecision in annotation. The idea was inspired by work on Transformation Based Learning.There are many differences between our method for correction and the standard TBL approach. On the one hand $(LP)^2$ does not use any templates for inducing rules, but it uses a very generic algorithm. Moreover rules in $(LP)^2$ are independent from each other and the starting seed for rule induction is chosen randomly. In classic TBL rules are ordered and the best seed for induction is selected at every cycle. Here the difference relies mainly in efficiency in induction, as some authors have shown that it is possible to redefine the TBL paradigm using rule independence and random seed selection, [Hep00]. Finally our correction concerns tag shifting, while Brill's correction relies on tag identity change. In our experience correction pays off in recognising slot fillers with a high degree of variability (e.g., the speaker in the CMU experiment, where it accounts for up to 7% of accuracy), while it has minimum relevance on highly standardised slot fillers (such as many slots in the Jobs task).

Finally it is interesting to note the way rule pattern length affects accuracy. Some slots are insensible to it (e.g. location), while others definitely need a longer pattern. This is important information to monitor as the window size strongly influences training time (see Figure 13).

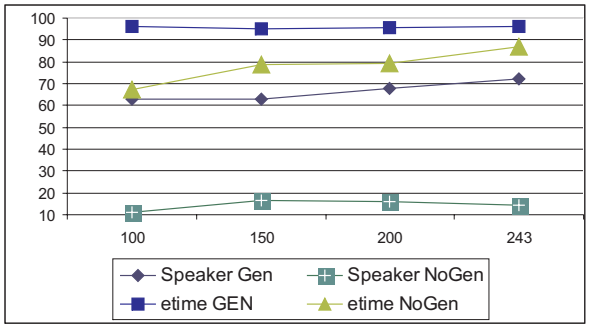

Figure 12: Effect of training data quantity in the CMU seminar task.The generalisation-based version converges immediately to the optimal value for <etime> and shows a positive trend for <speaker>. The non NLP based version shows overfitting on <speaker> and slowly converges on <etime>)

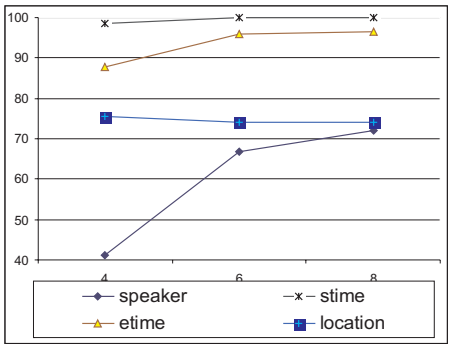

Figure 13: The effect of different window sizes in learning annotation rules.Some fields are not affected (e.g. location), while others (speaker) are sensitive to window size.

Table 4: Comparison between the NLP-based generalisation version $(\mathrm{LP})^2_G$ with the version without generalisation $(\mathrm{LP})^2_{NG}$

Slot	$(\mathrm{LP})^2_G$	$(\mathrm{LP})^2_{NG}$
speaker	72.1	14.5
location	74.1	58.2
stime	100	97.4
etime	96.4	87.1
All slots	89.7	78.2

4.1 NLP-based Generalisation

$(LP)^2$ uses shallow NLP processing for generalisation in order to reduce data sparseness: it allows capturing some general aspects beyond the simple flat word structure. Morphology allows overcoming of data sparseness due to number/gender word realisations (an aspect very relevant in morphologically rich languages such as Italian), while POS tagging information allows generalisation over lexical categories. Gazetteer and Named Entity Recognition information allow generalizing over some (user-defined) semantic classes, etc.

In principle such types of generalisation produce rules of better quality than those matching the flat word sequence; rules that tend to be more effective on unseen cases. This is because they are generic processes performing equally well on unseen cases; therefore rules relying on their results apply successful on unseen cases. This intuition was confirmed experimentally: we compared the rules produced by two versions of $(LP)^2$, with and without NLP-based generalisation (but with capitalisation information). The latter ($(\mathrm{LP})^2_{NG}$) learned a set of low coverage rules covering sparse data, while the first ($(\mathrm{LP})^2_G$) induced quite a number of high quality generic rules covering a large number of cases and a reduced number of rules for some exceptions (Figure 14 and Table 5). Moreover $(\mathrm{LP})^2_G$ definitely outperforms $(\mathrm{LP})^2_{NG}$ on the test corpus (Table 4), while having comparable results on the training corpus. NLP reduces the risk of overfitting the training examples. Figure 12 shows experimentally how $(\mathrm{LP})^2_G$ avoids overfitting on the speaker field (constant rise in f-measure when the training corpus is increased in size), while $(\mathrm{LP})^2_{NG}$ present a clear problem of overfitting (reduction of effectiveness when more examples are provided). Producing more general rules also implies that the covering algorithm converges more rapidly because its rules tend to cover more cases. This means that $(\mathrm{LP})^2_G$ needs less examples in order to be trained, i.e., rule generalisation also allows reducing the training corpus size. Figure 12 also shows how on etime $(\mathrm{LP})^2_G$ is able to converge at optimal values with 100 examples only, while accuracy in $(\mathrm{LP})^2_{NG}$ slowly increases with more training material, even if it is not able to reach the same accuracy as $(\mathrm{LP})^2_G$, even when using half of the corpus for training. Not surprisingly the role of shallow NLP in the reduction of data sparseness is more relevant for semi-structured or free texts (e.g. the CMU seminars) than on highly structured documents (e.g. HTML pages).

Table 5: Another Comparison between the NLP-based generalisation version $(\mathrm{LP})^2_G$ with the version without generalisation $(\mathrm{LP})^2_{NG}$

	$(\mathrm{LP})^2_G$	$(\mathrm{LP})^2_{NG}$
Average rule coverage	10.2	6.2
Selected rules	887	1136
Rules covering 1 case	133 (14%)	560 (50%)
Rules covering >50 cases	37	15
Rules covering >100 cases	19	3

4.2 Robustness

In order to be able to cope with a number of text types (mixed types included), it is necessary to be able to select the reliable level of linguistic analysis for the specific task at hand. $(LP)^2$ is able to do it at a very sophisticated degree. As a matter of fact the best strategy for each information or context is learnt separately. This is because it is able to decide that using the POS tagger information is a good strategy for recognizing - say - the speaker but not the best strategy to spot the seminar location or starting time. This claim was proven in an experiment where texts written in mixed Italian/English were used. They were seminar announcement taken from the ITC's (www.itc.it) seminar mailing list; the task was to extract title of seminar, location, date, time and speaker. The texts were mixed in a pseudo-random mixed Italian/English: the body announcing the seminar (containing location, date and time) was **generally** in Italian, but abstract and title of seminars were **generally** in English. Other parts were in either one of the two languages. An example is in Figure 15. We used an English preprocessor that was unreliable on the Italian part of the input. The preprocessor included a part of speech tagger, a gazetteer and named entity recogniser. We studied the effectiveness of three different versions of the algorithm: one using no NLP, one using just the POS tagger and one using the whole preprocessor. As expected, the more linguistic information is given, the greater the effectiveness, even if the linguistic information for the Italian parts was unreliable (Table 4.2).

The use of the POS helps consistently in recognising the date (+2.8%) and slightly the speaker (+1.3%); there is also a slight degradation on title (-0.4%), the latter being probably noise due to overgeneralisation in using the POS (as recall is higher and precision is lower). The improvement on dates is probably due to the ability to recognise numbers and therefore to generalize over them. As expected the use of gazetteer and NER contributes heavily in recognising the speaker (+9%) and there is some gain on both date (where there is some contributions by the NER) and location, probably a side effect due to the use of contextual rules (the NERC is very unreliable in recognising names of Italian meeting rooms). The accuracy in recognizing title is quite low; this is due to the fact that a title is a quite difficult object to recognize. In principlpe it is just a sequence of words, sometimes capitalised, sometimes uppercase and centered

Table 6: Comparison of accuracy on mixed Italian English seminar announcement (training on 75 texts, test on 73) . The first column refers to the system using only information on words and capitalization, the second with words capitalization and POS tags, the third with the full preprocessor including gazetteer lookup and Named Entity Recognition.

Slot	Words	POS	Full
speaker	74.1	75.4	84.3
title	62.8	62.4	62.8
date	90.8	93.4	93.9
time	100.0	100.0	100.0
location	95.0	95.0	95.5

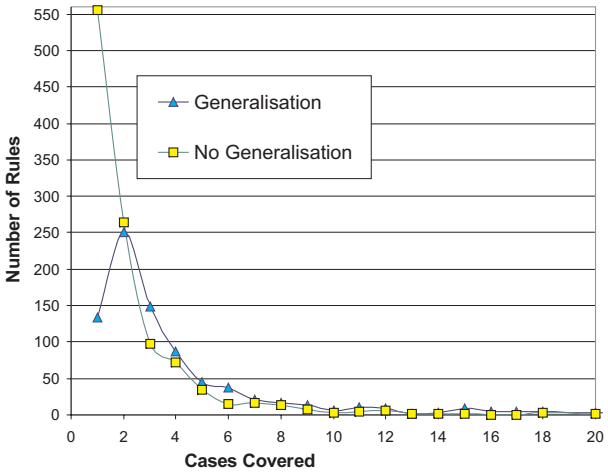

Figure 14: The effect of generalisation in reducing data sparseness.Number of rules (on y) covering number of cases (on X: we show up to 20 cases)

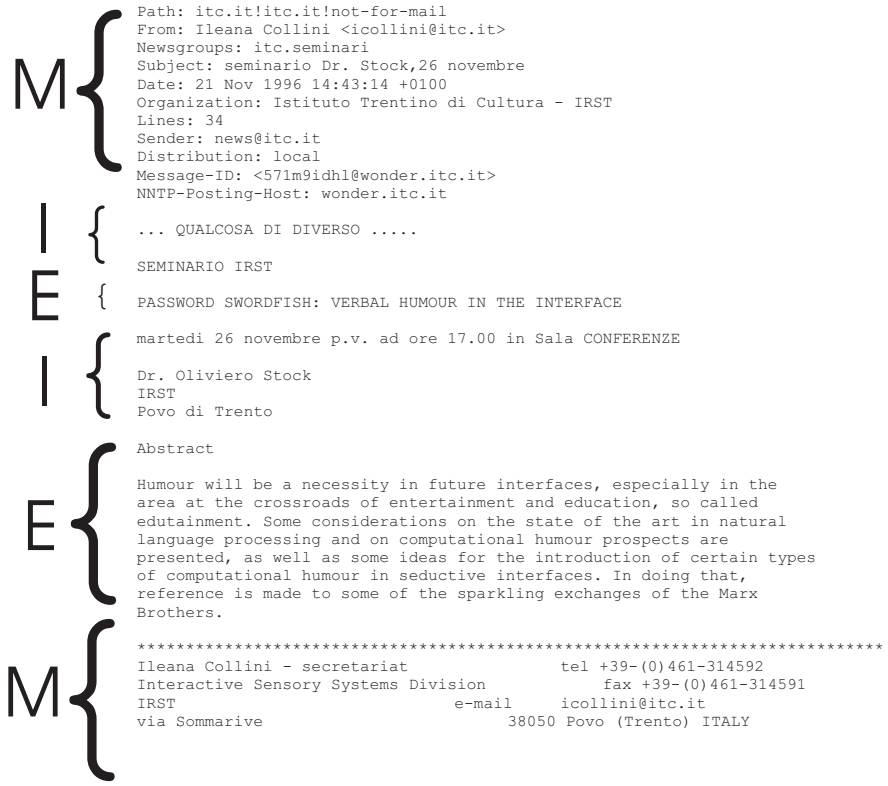

Figure 15: An example of mixed Italian/English seminar announcement."M" marks the parts in mixed language, while "E" shows the parts in English and "I" those in Italian.

around using spaces. Moreover its length can be quite variable. The only way to recognize it is using positioning in text and contextual information (it is often located near the speaker). This is why there is no increase in accuracy in adding linguistic information to the rules.

In our opinion this shows that the algorithm is able to make the best use of the linguistic information it uses. This means that rules using the unreliable NLP information were automatically discarded when necessary, while those using reliable information where kept. The measure of reliability was - for the learner - the ability to extract information, not linguistic competence.

5 Conclusion and Future Work

In this paper we have presented $(LP)^2$, an algorithm for learning to extract implicit events from documents of different types. We have described the algorithm and described experiments where the algorithm reaches excellent results.

We have also discussed how the different features of the algorithm contribute to such results. In particular we have focused on the contribution of linguistic information, showing that their use is not detrimental to the quality of results when such information is unreliable.

$(LP)^2$ has become the basis for two adaptive IE systems. The first one, called LearningPinocchio, has become a commercial system with applications developed and licenses released for further commercial use. Such applications are described in details in [CL03]. We have obtained the best experimental results on the CMU seminar announcement with it.

Amilcare is a system for information extraction developed as part of the AKT project (www.aktors.org) and has been mainly used for annotation for Knowledge Management and the Semantic Web. Currently it has been adopted as support to active annotation by two popular annotation tools for the Semantic Web: Ontomat, developed at the university of Karlsruhe[HSC02], and MnM, developed at the Open University [VVMD$^+$02]. It has also been integrated in Melita, an annotation tool for assisted corpus annotation developed in Sheffield [CDPW02]. In all these tools, Amilcare monitors the tags inserted by a user annotating a corpus and learns how to reproduce it. When the system has reached a reasonable accuracy in annotation, it starts helping users in the annotation process by providing every new text with a preliminary annotation to be corrected/confirmed by the user. Experiments have shown that in some cases Amilcare is able to save up to 70% of the user annotation after having trained on 2 or 3 dozens of short texts [CDPW02]. Amilcare is fully described in [Cir03]. Amilcare is also used in the Armadillo system [CDGW03] as support for unsupervised annotation of information on large repositories. Armadillo uses the redundancy of information in large distributed repositories (e.g. the Web) for bootstrapping learning on specific pages; the learner used is Amilcare. Within Armadillo, Amilcare has been tested on highly structured pages such as those produced by Google and semi-structured ones, e.g. manually compiled bibliographies. The use of Amilcare increases recall in discovering names of people working in a specific computer science department by 100% and increases the amount of bibliographic references retrieved from personal publication lists by 50% [CDGW03]. Finally Amilcare has been used by members of the scientific community as baseline for testing further IE algorithms and methodologies[MK03, FK03, SPSS03]. To date, the system as been released to about fifty users; half of them are commercial enterprises. Amilcare is freely available for use for research purposes. Instruction for obtaining a copy can be found at http://nlp.shef.ac.uk/amilcare.

Concerning future work, the use of linguistic information has been so far limited to shallow linguistic analysis, i.e. up to named entity recognition. We have also introduced the use of ontological reasoning, so that the system is able to reason over a hierarchy of semantic types during rule induction. We are currently working to include chunking and the use of Wordnet. The difficulty in using further level of generalisation is controlling overgeneralisation. In particular the extension to use Wordnet requires a careful monitoring. This is because the polysemy is very high in Wordnet and the generalisation derived (i.e. ex-

tending the condition in a rule not only to a specific word, but also to all the synonyms in the synsets) risks to overgeneralise if not kept under control, as explained in [CM97]. Current strategies under consideration are (1) accepting a synset only if there is multiple evidence of the reliability of the extension and (2) use a simplified version of Wordnet with reduced polysemy.

Finally, $(LP)^2$ uses a fairly simplistic algorithm from a machine learning point of view. More sophisticated algorithm is used, for example, in BWI (which uses boosting[SS88]). An extension of BWI to using linguistic features as in $(LP)^2$ is under development as part of the dot.kom project (www.dot-kom.org).

Acknowledgments

I developed the naive version of $(LP)^2$ and LearningPinocchio at ITC-Irst, Centro per la Ricerca Scientifica e Tecnologica, Trento, Italy. LearningPinocchio is property of ITC-Irst, see http://tcc.itc.it/research/textec/tools-resources/learningpinocchio.html. The advanced version of $(LP)^2$ and Amilcare have been developed at the University of Sheffield in the framework of the AKT project (Advanced Knowledge Technologies, http://www.aktors.org), an Interdisciplinary Research Collaboration (IRC) sponsored by the UK Engineering and Physical Sciences Research Council (grant GR/N15764/01) (www.aktors.org). The development of Amilcare has been also supported by the dot.kom project (IST-2001-34038) funded by the European Commission in Framework 5 (www.dot-kom.org).

This paper has been produced using both material from papers I have written in the past [Cir01a, Cir01b, CL03], and also some new material. Section 1 and the first two paragraphs of section 2 were written in cooperation with Alberto Lavelli.

References

[AHB+93] D. Appelt, J. Hobbs, J. Bear, D. Israel, and M. Tyson. FASTUS: A finite-state processor for information extraction from real-world text. *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 1172–1178, 1993.

[CDGW03] Fabio Ciravegna, Alexiei Dingli, David Guthrie, and Yorick Wilks. Integrating information to bootstrap information extraction from web sites. In *Proceedings of the IJCAI 2003 Workshop on Information Integration on the Web, workshop in conjunction with the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, 2003. Acapulco, Mexico, August, 9-15.

[CDPW02] Fabio Ciravegna, Alexiei Dingli, Daniela Petrelli, and Yorick Wilks. User-system cooperation in document annotation based on information extraction. In *Proceedings of the 13th International*

Conference on Knowledge Engineering and Knowledge Management, EKAW02. Springer Verlag, 2002.

[Cir01a] Fabio Ciravegna. Adaptive information extraction from text by rule induction and generalisation. In *Proceedings of 17th International Joint Conference on Artificial Intelligence (IJCAI)*, 2001. Seattle.

[Cir01b] Fabio Ciravegna. $(LP)^2$, an adaptive algorithm for information extraction from web-related texts. In *Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining held in conjunction with the 17th International Joint Conference on Artificial Intelligence*, 2001. Seattle, http://www.smi.ucd.ie/ATEM2001/.

[Cir03] Fabio Ciravegna. Designing adaptive information extraction for the Semantic Web in Amilcare. In S. Handschuh and S. Staab, editors, *Annotation for the Semantic Web*, Frontiers in Artificial Intelligence and Applications. IOS Press, 2003.

[CL03] Fabio Ciravegna and Alberto Lavelli. Learningpinocchio: Adaptive information extraction for real world applications. *Journal of Natural Language Engineering*, 2003. in press.

[CM97] M. Califf and R. Mooney. Relational learning of pattern-match rules for information extraction. In *Working Papers of the ACL-97 Workshop in Natural Language Learning*, pages 9–15, 1997.

[Dou98] A. Douthat. The message understanding conference scoring software user's manual. In *Proceedings of the 7th Message Understanding Conference*, 1998. $www.itl.nist.gov/iaui/894.02/related_projects/muc/$.

[FK00] D. Freitag and N. Kushmerick. Boosted wrapper induction. In R. Basili, F. Ciravegna, and R. Gaizauskas, editors, *ECAI2000 Workshop on Machine Learning for Information Extraction*, 2000. www.dcs.shef.ac.uk/ fabio/ecai-workshop.html.

[FK03] Aidan Finn and Nicholas Kushmerick. Active learning selection strategies for information extraction. In *Proccedings of the Workshop on Adaptive Text Extraction and Mining*. held in conjunction with the 14th European Conference on Machine Learning and the 7th European Conference on Principles and Practce of knowledge Discovery in Databases, September 2003.

[FM99] D. Freitag and A. McCallum. Information extraction with hmms and shrinkage. In *AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999.

[Fre98a] D. Freitag. Information extraction from html: Application of a general learning approach. *Proceedings of the Fifteenth Conference on Artificial Intelligence AAAI-98*, pages 517–523, 1998.

[Fre98b] D. Freitag. Multistrategy learning for information extraction. *Proceedings of ICML-98*, 1998.

[Gri97] Ralph Grishman. Information extraction: Techniques and challenges. In Maria Teresa Pazienza, editor, *Information Extraction: a multidisciplinary approach to an emerging technology*, 1997.

[Hep00] Mark Hepple. Independence and commitment: Assumptions for rapid training and execution of rule-based part-of-speech taggers. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, 2000. Hong Kong.

[HGA+98a] K. Humphreys, R. Gaizauskas, S. Azzam, C. Huyck, B. Mitchell, H. Cunningham, and Y. Wilks. Description of the university of sheffield lasie-ii system as used for muc. In *Proc. of the 7th Message Understanding Conference*, 1998. *www.itl.nist.gov/iaui/*894.02/*related*$_p$*rojects/muc/*.

[HGA+98b] Kevin Humphreys, Robert Gaizauskas, Saliha Azzam, Chris Huyck, Brian Mitchell, Hamish Cunningham, and Yorick Wilks. Description of the university of sheffield LaSIE-II system as used for muc. In *Proc. of the 7th Message Understanding Conference*, 1998. *www.itl.nist.gov/iaui/*894.02/*related*$_p$*rojects/muc/*.

[HSC02] S. Handschuh, S. Staab, and F. Ciravegna. S-CREAM - Semi-automatic CREAtion of Metadata. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW02*. Springer Verlag, 2002.

[KWD97] N. Kushmerick, D. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1997.*, 1997.

[MCF+98] S. Miller, M. Crystal, H. Fox, L. Ramshaw, R. Schwartz, R. Stone, and R. Weischedel. Bbn: Description of the SIFT system as used for MUC7. In *Proceedings of the 7th Message Understanding Conference*, 1998. *www.itl.nist.gov/iaui/*894.02/*related*$_p$*rojects/muc/*.

[MK03] David Masterson and Nicholas Kushmerick. Information extraction from multi-document threads. In *Proccedings of the Workshop on Adaptive Text Extraction and Mining*. held in conjunction with the 14th European Conference on Machine Learning and the 7th European Conference on Principles and Practce of knowledge Discovery in Databases, September 2003.

[MMJL86] R. S. Mickalski, I. Mozetic, Hong J., and H. Lavrack. The multi purpose incremental learning system aq15 and its testing application to three medical domains. In *Proceedings of the 5th National Conference on Artificial Intelligence*, 1986. Morgan Kaufmann publisher.

[MMK98] I. Muslea, S. Minton, and C. Knoblock. Wrapper induction for semistructured web-based information sources. In *Proceedings of the Conference on Automated Learning and Discovery (CONALD), 1998.*, 1998.

[MUC98] MUC7. *Proceedings of the 7th Message Understanding Conference (MUC7).* Nist, 1998. $www.itl.nist.gov/iaui/894.02/related_projects/muc/$.

[Sod99] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1):233–272, 1999.

[SPSS03] Georgios Sigletos, Georgios Paliouras, Constantine D. Spyropoulos, and Takis Stamatopoulos. Meta-learning beyond classification: A framework for information extraction from the web. In *Proccedings of the Workshop on Adaptive Text Extraction and Mining.* held in conjunction with the 14th European Conference on Machine Learning and the 7th European Conference on Principles and Practce of knowledge Discovery in Databases, September 2003.

[SS88] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, 1988.

[VVMD+02] M. Vargas-Vera, Enrico Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna. MnM: Ontology driven semi-automatic or automatic support for semantic markup. In *Proc. of the 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW02.* Springer Verlag, 2002.