Designing Adaptive Information Extraction for the Semantic Web in Amilcare

Fabio Ciravegna and Yorick Wilks
Department of Computer Science, University of Sheffield,
Regent Court, 211 Portobello Street,
Sheffield, S1 4DP, UNITED KINGDOM

Abstract.

1   Introduction

A crucial aspect of creating the Semantic Web (SW) is to enable users to create machine-readable Web content. Emphasis in the research community has till now been put on building tools for manual annotation of documents (e.g. [18]), but only recently has the problem of producing automatic or semi-automatic methods for annotating documents become an issue [17, 27]. The main problem with manual annotation is that it is a difficult, slow, time-consuming and tedious process that involves high costs and very often a large number of errors as well (up to 25% in some cases [24]). The latter is especially true in case of user with no experience of document annotation (naive annotators), while it slightly improves with expert annotators (about 15%). Manual annotation of documents by naive Web users is quite unlikely to be correct or even performed at all. The semantics of ontologies (generally used as reference for annotation) can be opaque to a layman, selecting the correct concepts or even the correct ontology could be out of reach for such users. Some type of support is needed to make the annotation process feasible. According to some researchers, the future of the SW will not rely on few large, complex, consistent ontologies shared by great numbers of users. The SW will instead be dynamic and based on a great number of small ontological components [19]. These components will be continuously extended, merged or created, and therefore the annotation services associated to them will have to be constantly adjusted or revised according to these changes. This poses a number of obvious constraints and requirements on the technology to support annotation in terms of usability, portability and maintainability.

Information Extraction from texts (IE) is an automatic method for locating important facts in electronic documents for successive use, e.g. for document annotation or for information storing (such as populating an ontology with instances). IE can provide support in document annotation either in an automatic way (unsupervised extraction of information) or semi-automatic way (e.g. as support for human annotators in locating relevant facts in documents, via information highlighting). IE was defined in the past years mainly as a technology created by the MUC conferences, which were comparative evaluations sponsored by DARPA[24]. The main requirements for the technology have been, in the MUC context, effectiveness with respect to quite complex tasks and reduced porting time by IE experts [9]. So far, only limited application to the Web has been attempted, especially for the Semantic

Web [21]. Its use has been generally intended as a technology providing generic support to annotation (e.g. recognising Named Entities [11]). When it has been used as support to more specific tasks, IE experts are generally involved in the development cycle in order to develop IE grammars or knowledge bases. As mentioned above, we believe that if we really want to use IE as support for annotation for the SW, we have to develop a new kind of technology where issues such as usability (e.g. by naive Web users), cost of new applications, portability and maintainability are the main design issues. Note that other application areas, such as Knowledge Management, are currently posing similar requirements to IE: interestingly, these are areas in which the SW could play (or is already playing) a major role [5].

Porting (and or maintaining) IE systems actually means coping with four (overlapping) main tasks:

- Adapting to new domain information: implementing (or modifying) lexica, knowledge bases, etc., designing new templates, so that the system is able to manipulate domain-specific concepts;

- Adapting to sublanguage features: modifying grammars and lexica in order to cope with specific linguistic constructions that are typical of the application;

- Adapting to different text genres: specific text genres (e.g. medical abstracts, scientific papers, police reports) may have their own lexis, grammar or discourse structure.

- Adapting to different text types: Web-based documents can vary from rigidly structured XML documents (e.g. tables) to free texts. Each different text type can have different requirements in term of language analysis.

IE experts are quite rare in industry and therefore, if IE is to be used, it must be adaptable using knowledge only of the domain (e.g. via text annotation only). This is quite a shift from the classic MUC-oriented technology (e.g. [20, 1]) which relies strongly on IE experts. Also, adaptive technology that requires linguistic (not domain) annotation (e.g. [23]) is not suitable in this context. Moreover, in terms of amount of annotated material for training, the need must be reduced in order to minimize the burden on the user side (lower cost, reduced development/maintenance time).

Concerning porting across text types, in classical Natural Language Processing (NLP) adapting to new text types has been generally considered as a task of porting across different types of free texts [3]. The use of IE for the SW requires an extension of the concept of text types to new, unexplored, dimensions. Linguistically-based methodologies used for free texts can be difficult to apply or ineffective on highly structured texts such as web pages produced by data bases. They are not able to cope with the variety of extralinguistic structures (e.g. XML tags, document formatting, and stereotypical language) that are used to convey information in such documents. On the other hand, wrapper-like algorithms designed for highly structured HTML/XML pages [22, 25] are largely ineffective on unstructured texts (e.g. free texts). This is because such methodologies make scarce (or no) use of NLP, avoiding any generalization over the flat word sequence, and tending to be ineffective on free texts, for example, because of data sparseness [6]. The challenge is to develope methodologies able to fill the gap between the two approaches in order to cope with different text types. This is particularly important for the SW, as Web pages can contain documents of any type and even a mix of text types, e.g., they can contain both free, semi-structured and structured texts at the same time (see Figure

Figure 1: The cnn.com page as example of mixed text types page

1 for an example). Even methodologies that require the selection of alternative grammars or methodologies for specific texts types (e.g. [11, 26]) are not suitable when a mixture of texts types is present, because the type mixture would rather require mixing such grammars, rather than selecting one of them; non-IE experts simply cannot merge grammars. The IE methodology for the SW must be adaptive also in this sense, of being able to recognise what linguistic resources are useful for analysing a specific context, sometimes even a context local to specific information: e.g. if a piece of information is in a table, a deep linguistic method relying on a parser's results is unlikely to be effective, but when the information is in a free text zone, the same method is likely to work). To our knowledge none of the classic IE methods is adaptive in this sense, but we believe that this is a main requirement for the application for the SW.

In this paper we present Amilcare, an adaptive IE system designed as support to document annotation in the SW framework. It is based on an adaptive methodology that meets most of the requirements mentioned above and in particular:(1) portability by a wide range of users, from naive users to IE experts; (2) ability to cope with different types of texts (including mixed ones); (3) possibility to be inserted in the usual user annotation environment providing minimum disruption to usual annotation activities; (4)portability with reduced number of texts. In this paper we describe Amilcare, the learning algorithm, the details of the human-system interaction. Finally we describe a number of experiments that show its effectiveness and its usability.

2 Amilcare

Amilcare is an adaptive IE system, i.e. it uses machine learning to adapt to new applications/domains. It is rule based, i.e. its learning algorithm induces rules that extract information.

Rules are learnt by generalising over a set of examples found in a training corpus annotated with XML tags. The system learns how to reproduce such annotation via Information Extraction. Amilcare is based on the $(LP)^2$ algorithm [4], a supervised algorithm that falls into a class of Wrapper Induction Systems (WIS) using LazyNLP [6]. Unlike other WIS [22, 25, 15, 16], LazyNLP WIS use linguistic information. They try to learn the best (most reliable) level of language analysis useful (or effective) for a specific IE task by mixing deep linguistic and shallow strategies. The learner starts inducing rules that make no use of linguistic information, like in classic wrapper-like systems. Then it progressively adds linguistic information to its rules, stopping when the use of linguistic information becomes unreliable or ineffective. Linguistic information is provided by generic NLP modules and resources defined once for all and not to be modified by users to specific application needs. Pragmatically, the measure of reliability here is not linguistic correctness (immeasurable by incompetent users), but effectiveness in extracting information using linguistic information as opposed to using shallower approaches. Unlike previous approaches where different algorithm versions with different linguistic competence are tested in parallel and the most effective is chosen [26], lazy NLP-based learners learn which is the best strategy for each information/context separately. For example they may decide that using parsing is the best strategy for recognising the speaker in a specific application on seminar announcements, but not the best strategy to spot the seminar location or starting time. This has shown to be very effective for analysing documents with mixed genres, e.g. web pages containing both structured and unstructured material, quite a common situation in web documents [8].

Amilcare can work in three modes: training mode, test mode and production mode. The **training mode** is used to induce rules, so to learn how to perform IE in a specific application scenario. Input in training mode is: (1)a scenario (e.g. an ontology in the SW); (2) a training corpus annotated with the information to be extracted. Output of the training phase is a set of rules able to reproduce annotation on texts of the same type. The **testing mode** is used to test the induced rules on an unseen tagged corpus, so to understand how well it performs for a specific application. When running in test mode Amilcare first of all removes all the annotations from the corpus, then re-annotates the corpus using the induced rules. Finally the results are automatically compared with the original annotations and the results are presented to the user. Output of the test phase is: (1) the corpus reannotated by the system; (2) a set of accuracy statistics on the test corpus: recall, precision and details on the mistakes the system does. Amilcare uses an internal scorer, but it is also compatible with some standard scorers such as the MUC scorer [13]. During testing it is possible to decide to retrain the learner with different system parameters in order to tune its accuracy (e.g. to obtain more recall and/or more precision). Tuning takes a fraction of time with respect to training. The **production mode** is used when an application is released. Amilcare annotates the provided documents. If a user is available to revise its results, the learner uses the user corrections to retrain. The training/test/production modes can actually be interleaved so to produce an active learning based annotation. In active learning [2] user annotation and system annotation are interleaved in order to minimize the amount of user annotation. As we will see in the following this greatly reduces the burden of document annotation.

Amilcare's default architecture includes the connection with Annie, Gate's shallow IE system [11] which performs tokenization, part of speech tagging, gazetteer lookup and named entity recognition. Any other preprocessor can be connected via the API. The preprocessor is also the only language-dependent module, the rest of the system being language independent (experiments where performed in English and Italian).

Condition	Additional Knowledge				Action
Word	Lemma	LexCat	Case	SemCat	Tag
the	the	det	low		
seminar	seminar	noun	low		
at	at	prep	low		
4		digit	low		<time>
pm		noun	low	timeid	
will	will	verb	low		

Figure 2: A tagging rule

2.1 The learning algorithm

$(LP)^2$ induces two types of symbolic rules in two steps: (1) rules that insert annotations in the texts; (2) rules that correct mistakes and imprecision in the annotations provided by (1). Rules are learnt by generalising over a set of examples marked via XML tags in a training corpus. 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 XML tag in the texts. Each rule inserts a single tag, e.g. `</speaker>`. As positive examples the rule induction algorithm uses XML annotations in a training corpus. 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 extracts 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 in the initial rule pattern, e.g. if the third word is "seminar", a condition word3="seminar" is created. Each initial rule is then generalised (see subsections 2.1.1) and the k best generalisations are kept: retained rules become part of the best rules pool. When a rule enters such pool, all the instances covered by the rule are removed from the positive examples pool, i.e. they will no longer be used for rule induction $((LP)^2$ is a sequential covering algorithm). Rule induction continues by selecting new instances and learning rules until the pool of positive examples is empty. Some tagging rules (contextual rules) use tags inserted by other rules. For example some rules will be used to close annotations, i.e. they will use the presence of a `<speaker>` to insert a missing `</speaker>`. In conclusion the tagging rule set is composed of both the best rule pool and the contextual rules. Tagging rules when applied on the corpus may 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. $(LP)^2$ induces rules for shifting wrongly positioned tags to the correct position. It learns from the mistakes made in tagging the training corpus. 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. The induction algorithm used for the best tagging rules is also used for shift rules: initial instance identification, generalisation, test and selection. Shift rules are accepted only if they report an acceptable error rate.

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

Condition					Action
Word	Lemma	LexCat	Case	SemCat	Tag
	at				
		digit			<time>
				timeid	

Figure 3: A generalised tagging rule

tag the texts. Then contextual rules are applied in a loop until no new tags are inserted, i.e. some contextual rules can match also tags inserted by other contextual rules. Then correction rules correct imprecision. Finally each tag inserted by the algorithm is validated. In many cases 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.

2.1.1   Rule Generalization

The initial rule pattern matches conditions on word strings as found in a window around each instance. This type of rules is the typical type of rules wrapper induction systems produce. We have previously shown [6] that these types of rules are suitable for highly structured texts, but quite ineffective on free texts, because of data sparseness implied by the high flexibility of natural language forms. It is therefore important to generalise over the plain word surface of the training example. The initial rule pattern is integrated with additional conditions on the results of the linguistic preprocessor. This means that conditions are set not only on the strings (word="companies"), but also on its lemma ("company"), its lexical category ("noun"), case information ("lowercase"), a list of user defined classes from a user-defined dictionary or a gazetteer and annotations by a named entity recogniser. Figure 2 summarizes the new form of the initial rule. The initial rule effectiveness is compared against a set of more general rules obtained through a general to specific beam search that starts modelling the annotation instance with the most general rule (the empty rule matching everything in the corpus) 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 (matching strings or part of the knowledge provided by the linguistic preprocessor). The specialization process stops when the generated rules have reached a satisfying accuracy and coverage and any further specialization does not improve precision. In this way the rules that incorporate different levels of linguistic knowledge (from deep linguistic to word matching) are put in direct competition and just the best combination of k rules survive. Figure 3 shows a possible generalization of the rule in figure 2. The choice of the level of linguistic knowledge is local to the specific instance we are trying to model (local constraints, e.g. the information is located in a table), but it also depends on the ability to generalize to cover other instances (global constraints, e.g. it is a noun followed by a capitalised word). For this reason the approach satisfies the requirement of coping with different types of texts including mixed types, because it is able to model different parts of texts in which different information is located using different (linguistic) strategies. Details of the algorithm can be found in [6].
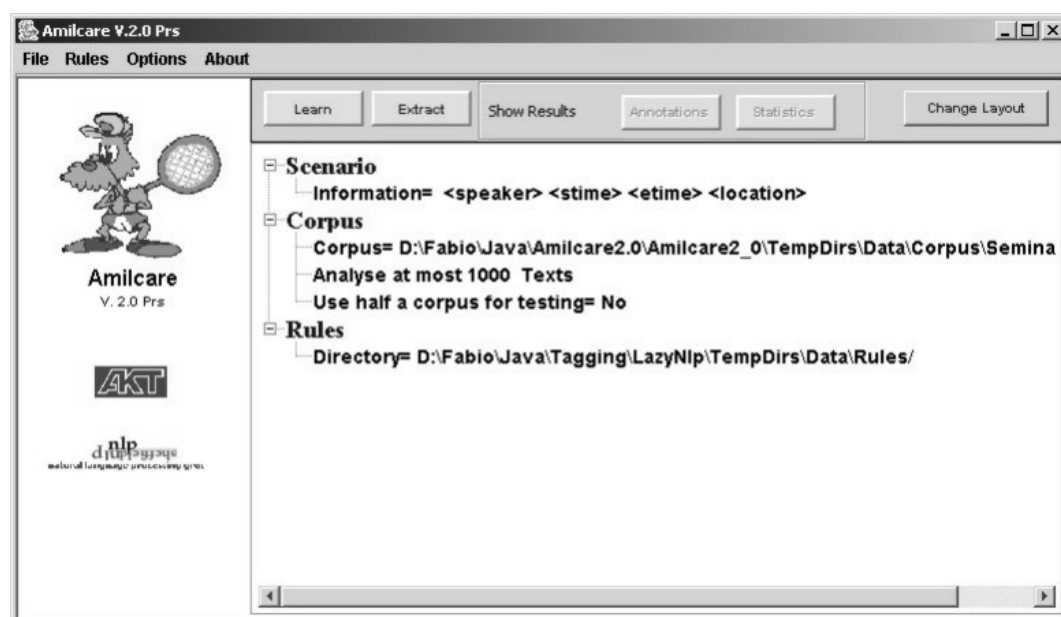
Figure 4: The interface for naive users

3 Different Views for Different Users

Amilcare provides different user types with different interaction methodologies so to exploit the knowledge of the user at hand to a maximum extent. In our experience in delivering adaptive IE systems we discovered that simplifying porting is not the main issue for the success of IE: it is necessary to enable different users with different capabilities to produce applications, whose complexity may be proportional to their knowledge of IE [8]. For this reason while Amilcare's naive users are just required to provide an annotated corpus and a list of tags (or ontology), IE experts can actually contribute to the IE grammars development. In this section we discuss the interaction methodology for three different user types: naive users, Amilcare experts and IE experts. Section 4.1 will actually discuss how Amilcare can be inserted in an existing annotation environment.

3.1 The Naive User's View

A naive (from the IE point of view) user will have knowledge of the domain, understand the texts, know where to extract information and will be able to provide XML tagged texts. XML tagged texts can be provided with a range of tools, from Alembic [12], to Melita [7], to MnM [27] or Ontomat [18]. Such tools provide intuitive interfaces that allow to annotate, ignoring the complexity of XML syntax. Amilcare can be trained on an annotated training corpus by using the interface shown in figure 4. The user is required to declare:

- the ontology used: this is a flat list of tags for many users (and in the example);

- the corpus to train on;

- the number of texts to be analysed within the corpus;

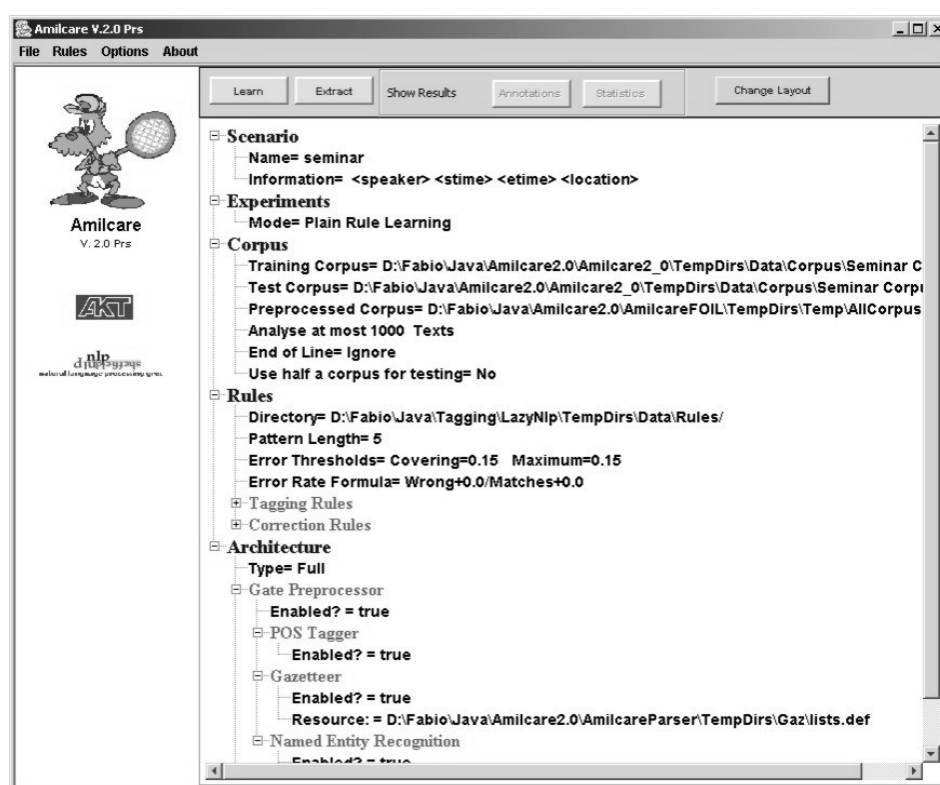- the directory where the induced rules will be stored.

Figure 5: The Interface for the expert developer allowing deep control over the system

Then Amilcare is ready for learning. Simply pressing the "Learn Rules" button starts the process. When learning is over, it is possible (and generally advisable) to test the system effectiveness on a set of unseen annotated texts before delivering the application. In the same interface, just indicate the test corpus and click "Apply Rules". It is then possible to inspect the IE effectiveness on the test corpus in two ways: (1) "Show Results" allows to visually inspect the texts as annotated by the system, see Figure 6; (2) "Show Statistics" shows the effectiveness of the extraction process expressed using the usual IE metrics: number of original user annotations (Possible), number of annotation actually produced by the system (Actual), number of correct/wrong system annotations, number of missed annotations, number of partially correct annotations (i.e. overlapping with the user annotation, but not precise). Finally precision, recall and f-measure are provided. It is also possible to inspect the list of all the correct, wrong, partial and missed matches in lists that compare each system annotation with the corresponding user annotation. If the results are satisfying, the application can be delivered. If it is not, the most obvious action is to increase the size of the training corpus. Defining an application in this way is quite simple and any user able to annotate texts is able also to define an IE application with a very limited ad hoc training. It is also generally possible to obtain good applications in this way.

3.2   *The Amilcare Expert View*

An intermediate user figure between the IE expert and the completely naive user is the Amilcare expert, i.e. a user not able to delve into the complexity of IE (e.g. developing rules), but able to optimise Amilcare's results with respect to the application needs, for example by better
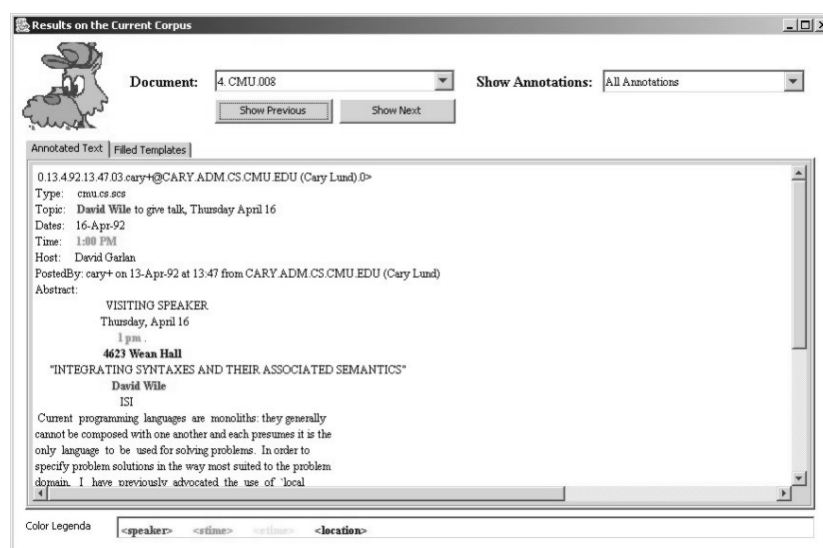
Figure 6: Results presentation of system's annotation for a text

controlling the experimental process. In this case it is possible to influence a number of system parameters by tuning some knobs (see Figure 5). For example it is possible to adjust: (1) the precision recall balance; (2) the rule pattern length (the longer the rule pattern, the better the effectiveness but the longer the training time); (3) the minimum rule selection conditions (error threshold and minimum number of matches), influencing therefore the global accuracy. The effect of changing values of such parameters can be learned by performing a number of trials. No IE knowledge is necessary, just detailed knowledge of the system behaviour and knowledge of how to perform experiments is required. The quality of the application produced by the Amilcare expert is generally better than the one produced by the naive user as it is better optimised. Note that the transition from naive to Amilcare expert can be done gradually by learning how to turn some knobs or to better organise the experiment, leaving the rest of the parameters untouched. For this reason it is possible to satisfy a range of users rather than just two types. The Amilcare's expert is the more likely user type expected in real world applications.

3.3 The IE Expert's View

Experience has shown that IE experts, when available, suffer if they are not enabled to contribute to the application development in an active way [8]. For this reason Amilcare provides a number of tools for IE experts. It provides two modes for application development: system driven and user driven. In system-driven development users provide Amilcare with a training corpus to train on. Then users can inspect the induced rules and revise them (see Figure 7). It is possible to inspect all the developed rules and their (1) coverage on the corpus: correct matches (places where the rule provides the same annotation the user provided) and wrong matches (spurious annotations or misplaced annotations) are shown; (2) error rate (approximately wrong matches divided by number of matches); (3) total number of matches on the corpus. It is then possible to modify, delete or copy rules. The rule editor (Figure 8), supports editing by providing menus of possible fillers for different conditions, information on matches (correct and wrong) of the rule under development, etc. The other possibility is to develop
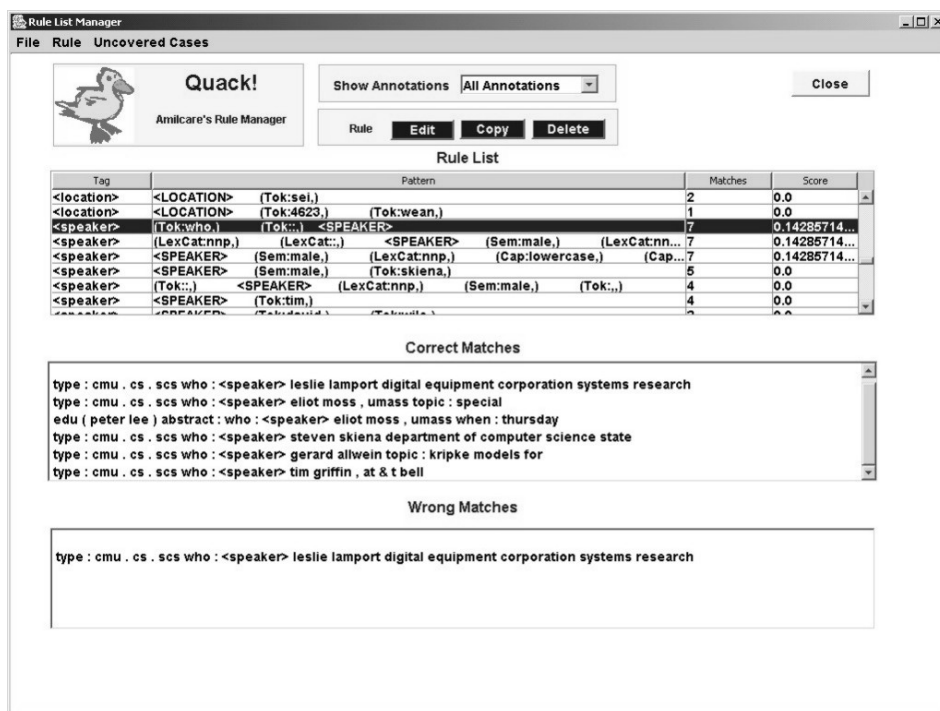
Figure 7: The Rule Manager enables the IE expert to inspect the grammar rules and their coverage

rules in a user-driven fashion. It is possible (1) to use the above mentioned strategy and editor and develop rules starting from scratch, then asking Amilcare to integrate/revise the manually developed rules by learning from the annotated corpus, (2) to use the Jape development environment included in Gate [11] to develop rules and then ask Amilcare to eventually integrate the rules. The Jape's and Amilcare rules are not compatible, but in this case Amilcare's rules will be applied afterwards and will correct/integrate the Jape rules results. The quality of the application developed by IE experts can be very high. The simple rule correction can considerably increase accuracy and reduce the need for a representative annotated corpus.

4   Amilcare as Support to Annotation

Amilcare can be used as active support to annotation for the SW. Different strategies can be used for integrating Amilcare with an Annotation Tool (AT). What we have seen so far is a batch mode, where Amilcare is controlled as a separate process via its own interface: it receives an annotated corpus in input and induces a set of rules, or it receives a corpus in input and returns a set of annotations (for rule testing/application). This is the most straightforward integration methodology. A more sophisticated integration can be obtained via the API, which allows full control on the system. It is possible to organise a batch interaction mode where users just interact with the AT, not with the IE system. To some extent the user could even ignore the presence of Amilcare behind the scenes. This is better organisation because the AT provides more suitable use(r)-specific interaction methodologies, being designed with a specific application in mind. Here the user annotates a batch of documents and the IE tool is trained on the whole batch. Then annotation is started on another batch and the IE system passes annotations to the interface which in turn proposes them to the annotating users. This is how Amilcare was integrated in MnM [27] and ONTOMAT [17]. In this section we will
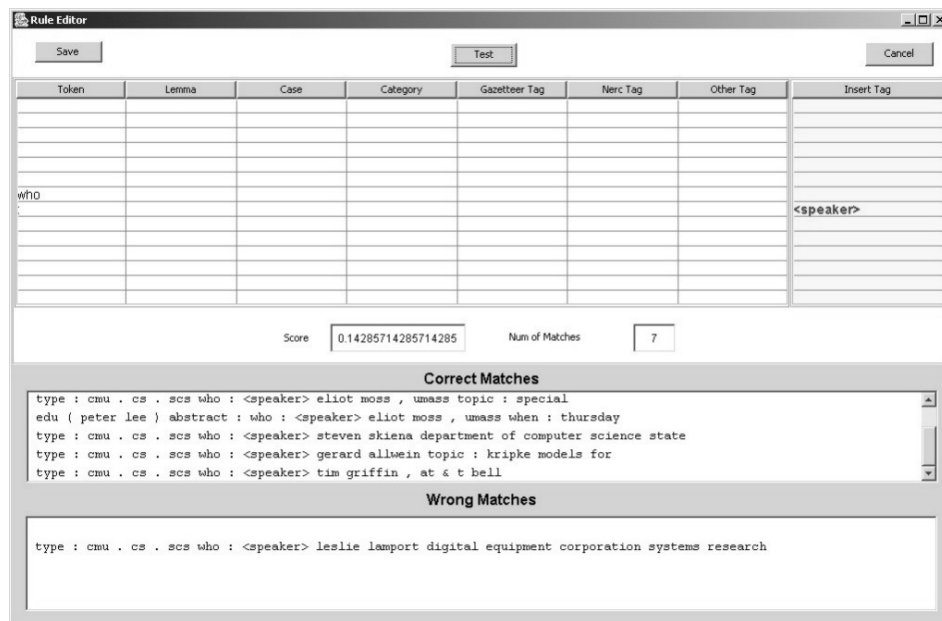
Figure 8: The Rule Editor allows to modify rules. It provides menus of possible fillers for the different fields, information on matches (correct and wrong) of the rule under development

present a more sophisticated strategy as it was implemented in Melita [7]. It allows to better exploit the potentialities of the IE technology.

4.1 Towards Sophisticated IE-based Annotation

The goal of the interaction between AT and IE system is both maximizing effectiveness of the annotation process and minimizing the burden of annotating/correcting on the user's sides. In the ideal initial scenario a user environment with a manual AT is already existent. The goal is to silently insert the IE-based support to annotation without asking the user to change his/her way of working. There are two main features of the interaction between users, AT and IE System (IES) that are considered: timeliness and intrusiveness. The first shows the ability to react to user annotation: how timely is the system to learn from user annotations. The latter represents the level to which the system bothers the user, because for example it requires CPU time (and therefore stops the user annotation activity) or because it suggests wrong annotations. The annotation process is split into two main phases from the IES point of view: (1) training and (2) active annotation with revision. In user terms the first corresponds to unassisted annotation, while the latter mainly requires correction of annotations proposed by the AT (upon suggestion from the IES). During *training*, users annotate texts without any contribution from the IES which is mainly using the user annotations to train its learner. We can define two sub-phases: (a) bootstrapping and (b) training with verification. During bootstrapping the only IES task is to learn from user annotations. This sub-phase ends when the system has induced a minimum set of rules. During the second sub-phase, the user continues with the unassisted annotation, but the IES behaviour changes, as it uses its rules to silently compete with the user in annotating the documents. The IES automatically compares its annotations with those inserted by the user and calculates its accuracy. Missing annotations or mistakes are used to retrain the learner. The training phase ends when the IES accuracy reaches the user preferred level of accuracy

(as declared by a user profile [7]). It is therefore possible to move to the next phase: active annotation. The active annotation with revision phase is heavily based on the IES suggestions and the user's main task is correcting and integrating suggested annotations (i.e. removing and adding annotations). Human actions are inputted back to the IES for retraining. This is the phase where the real system-user cooperation takes place: the system helps users in annotating; users feed back mistakes to help the system perform better. In user terms this is where the added value of the IES becomes apparent, because it heavily reduces the amount of annotation to insert manually. This supervision task is much more convenient from both cognition and actions. Correcting annotations is simpler than annotating bare texts, it is less time consuming and it is also likely to be less error prone.

The design of the interaction model aims to limit intrusiveness of the IES in a number of ways. First of all the IES does not require any specific AT or any specific adaptation by the user. It integrates in the usual user environment and provides suggestions through the AT in a way that is both familiar and intuitive to users. Secondly intrusiveness as a side effect of proactivity is coped with, especially during active annotation with revision, when the IES can bother users with unreliable annotations. The requirement here is to enable users to tune the IES behaviour so that the level of suggestions is appropriate. Notice that the acceptable level of intrusiveness is subjective: some users might like to receive suggestions largely regardless from their correctness, while others do not want to be bothered unless suggestions are absolutely reliable. Amilcare's internal "knob" tuning methods (e.g. to balance precision and recall) are too complex for most users, as they are designed for experts. In Melita the AT bridges the qualitative vision of users (e.g. a request to be more/less active or accurate) with the specific IES settings (e.g. change error thresholds), as also suggested in [10]. This is important because as mentioned the AT is designed for specific user classes and therefore able to elicit tuning requirements with the correct terminology for the specific context. Finally the IES training requires CPU time and this can slow down or even stop the user activity. For this reason most of the current systems use a batch mode of training so to limit training to specific moments (e.g. coffee time). In Melita background learning provides timely support without intrusiveness. The IES works in the background with two parallel and asynchronous processes. While the user annotates document n+1 the system learns the annotations inserted in document n (i.e. the last annotated). At the same time (i.e. as a separate process) the IES applies the rules induced in the previous learning sessions (i.e. from document 1 to document n-1) in order to annotate document n+1 (either for suggesting annotations during active annotation or in order to silently test its accuracy during unassisted learning). The advantage is that there is no idle time for users, as the manual annotation of a document generally requires a great deal more time than training on a single text and in any case the background learning can be performed by a low priority process. Timeliness is not fully obtained with the above interaction methodology: the IES annotation capability always refers to rules learned by using the entire annotated corpus but the last document. This means that the IES is not able to help when two similar documents are annotated in sequence. From the user point of view such a situation is equivalent to train on batches of two texts. In this respect the collaboration between the system and the user fails in being effective. We believe that timeliness is a matter of perception from the user side, not an absolute feature; therefore the only important matter is that users perceive it. When the order of document annotation is unimportant, it is possible to organize annotation so to avoid presenting similar documents in sequence (and hence hiding the small lack of timeliness). Melita uses a simple measure of similarity among texts: it runs Amilcare on the unannotated corpus; inserted annotations mean similarity of texts with respect to the part of the

corpus annotated so far, no inserted annotation means actual difference. Such information is used to make timeliness more effective: a completely uncovered document is always followed by a fairly covered document. In this way difference between successive documents is very likely and therefore the probability that similar documents are presented within the batch of two (i.e. the system's blindness window) is very low. Incidentally this strategy also tackles another major problem in annotation, i.e. user boredom, which can make user productivity and effectiveness fall proportional to time. Presenting users with radically different documents avoids the boredom that comes from coping with very similar documents in sequence.

5   Evaluating IE's contribution

We performed a number of experiments for demonstrating how fast the IES can converge to an active annotation status and to quantify its contribution to the annotation task, i.e. its ability to suggest correctly. We selected the CMU seminar announcements corpus, where 483 emails are manually annotated with speaker, starting time, ending time and location of seminars, [14]. Such corpus was already used for evaluating a number of adaptive algorithms, [14, 2, 16, 15, 4]. In our experiment the annotation in the corpus was used to simulate human annotation. We have evaluated the potential contribution of the IE system at regular intervals during corpus tagging, i.e. after the annotation of 5, 10, 20, 25, 30, 50, 62, 75, 100 and 150 documents (each subset fully including the previous one). Each time we tested the annotation accuracy on the following 200 texts in the corpus (so when training on 25 texts, the test was performed also on the following 25 texts that will be used for training on 50). The ability to suggest on the test corpus was measured in terms of precision and recall. Recall represents here an approximation of the probability that the user receives a suggestion in tagging a new document. Precision represents the probability that such suggestion is correct. The maximum gain comes in annotating stime and etime. This is not surprising as they present quite regular fillers. After training on only 20 texts, the system is potentially able to propose 368 stimes (out of 491), 303 are correct, 18 partially correct, 47 wrong, leading to Precision=84 Recall=61 (see Table 5). With 30 texts the recognition reaches P=91, R=78, with 50 P=92, R=80. The situation is very similar for etime, while it is more complex for speaker and location, where recall grows slowly: after 75 texts the system reaches P=96, R=60 on location and P=90, R=50 on speaker. This is due to the fact that locations and speakers are much more difficult to learn than time expressions because they are much less regular. We performed the same type of analysis on other corpora such as the Austin TX Jobs announcement corpus [2] and found similar results. Our experiments show that the contribution of the IES can be quite high. Reliable annotation can be obtained with limited training, especially when adopting high precision IES configurations. In the case of the CMU corpus, our experiments show that it is possible to move from bootstrapping to active annotation after annotating some dozens of texts. This shows that the IES contribution heavily reduces the burden of manual annotation and that such reduction is particularly relevant and immediate in case of quite regular information (e.g., time expressions). In user terms this means that it is possible to focus the activity on annotating more complex pieces of information (e.g. speaker), avoiding to be bothered with repetitive ones (such as stime). With some more training cases the IES is also able to contribute in annotating more complex cases.

Tag	Amount of Texts needed for Training	Precision	Recall
etime	20	96	72
stime	30	91	78
location	30	82	61
speaker	25	91	35

Table 1: Support to annotation after training on some dozens of texts.

6   Conclusion and Future Work

In this paper we have presented Amilcare, an adaptive IE system that is specifically designed to provide annotation for the SW. We believe that Amilcare is particularly well suited for the use in the SW because it enable even naive users to rapidly develop an IE-based support to annotation for specific SW services. This meets the requirements stated in the introduction for the use in a very dynamic SW scenario in which a great number of small ontological components and services are continuously developed and therefore the need for new (or improved) services to support annotation is constant. This will enable SW actors to develop (or extend/change) their existing services without requiring the constant presence of IE experts. We have also shown that the IE contribution to annotation can be quite high even with a limited number of training cases, relieving users from the burden of most of the annotation (even that used to actually train the IES!) since the very beginning. We believe that these results will help diffusing the use of IE for the SW: to date Amilcare has been released to a number of both academic and commercial institutions as support to either IE or annotation and it has been integrated in both ONTOMAT and MnM.

Two main issues are left for future work: the actual support to annotation (IE effectiveness) and the actual Amilcare's expressiveness. It is well known that IE systems reach 100% accuracy only in very simple cases. For this reason we definitely need to provide semi-automatic annotation services in which users producing SW documents are responsible to revise the IE suggestions. If the goal is to provide support for annotation for naive SW users (e.g. pet shop owners in [19]), imprecision could be missed in the revision process, greatly undermining the quality of annotation. At the present there is no solution other than continuing working on IE models to improve effectiveness. On the other hand the only alternative solution to IE is manual annotation: we believe that revising annotation is far easier than annotating from scratch, so the benefit of using IE is still there. Moreover the IE suggestions can be used to implicitly suggest users with a kind of standardisation of annotations (i.e. it is equivalent to another user suggesting); this is a proven way for helping in human annotation accuracy, used in many scientific exercises.

The second issue concerns Amilcare's expressiveness. There is no doubt that Amilcare is much more limited than full IE systems such as [20, 1]. For example it is able to spot information, but not to relate it (e.g. solving coreferences). This is because the goal was to provide a system very easy to use and port in the first instance. This can be a limitation in some applications [17]. We are currently working on a new version of Amilcare performing full IE. The main problem is not implementing a full IE system, but in making it usable by naive users: Amilcare is already inserted in Annie, therefore a methodology for a FASTUS-like approach

[1] to IE is potentially available. Our goal is to enable also naive users to develop full IE applications and this implies further studies on and new models of user-system interactions.

Acknowledgements

References

[1] APPELT, D., HOBBS, J., BEAR, J., ISRAEL, D., AND TYSON, M. Fastus: A finite-state processor for information extraction from real-world text. *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)* (1993), 1172–1178.

[2] CALIFF, M., AND MOONEY, R. Relational learning of pattern-match rules for information extraction. *Working Papers of the ACL-97 Workshop in Natural Language Learning* (1997), 9–15.

[3] CARDIE, C. Empirical methods in information extraction. *AI Journal 18*, 4 (1997), 65–79.

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

[5] CIRAVEGNA, F. Challenges in information extraction from text for knowledge management. *IEEE Intelligent Systems and Their Applications 27* (2001), 97–111. November.

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

[7] CIRAVEGNA, F., DINGLI, A., PETRELLI, D., AND WILKS, Y. User-system cooperation in document annotation based on information extraction. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW02* (2002), Springer Verlag.

[8] CIRAVEGNA, F., AND LAVELLI, A. Learningpinocchio: Adaptive information extraction for real world applications. In *Proceedings of 3rd Romand Workshop* (July 2001). Frascati, Italy.

[9] CIRAVEGNA, F., LAVELLI, A., AND SATTA, G. Bringing information extraction out of the labs: the pinocchio environment. In *Proceedings of the 14th European Conference on Artificial Intelligence* (2000), IOS Press. Berlin.

[10] CIRAVEGNA, F., AND PETRELLI, D. User involvement in adaptive information extraction: Position paper. 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/.

[11] CUNNINGHAM, H., MAYNARD, D., TABLAN, V., URSU, C., AND BONTCHEVA, K. "developing language processing components with GATE", 2002. www.gate.ac.uk.

[12] DAY, D., ABERDEEN, J., HIRSCHMAN, L., KOZIEROK, R., ROBINSON, P., AND VILAIN, M. Mixed-initiative development of language processing systems. In *Proceedings of the Fifth Conference on Applied Natural Language Processing* (1997).

[13] DOUTHAT, A. 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/.

[14] FREITAG, D. Multistrategy learning for information extraction. *Proceedings of ICML-98* (1998).

[15] FREITAG, D., AND KUSHMERICK, N. Boosted wrapper induction. In *ECAI2000 Workshop on Machine Learning for Information Extraction* (2000), R. Basili, F. Ciravegna, and R. Gaizauskas, Eds. www.dcs.shef.ac.uk/ fabio/ecai-workshop.html.

[16] FREITAG, D., AND MCCALLUM, A. Information extraction with hmms and shrinkage. In *AAAI-99 Workshop on Machine Learning for Information Extraction* (1999).

[17] HANDSCHUH, S., STAAB, S., AND CIRAVEGNA, F. S-CREAM - Semi-automatic CREAtion of Metadata. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW02* (2002), Springer Verlag.

[18] HANDSCHUH, S., STAAB, S., AND MAEDCHE, A. CREAM — Creating relational metadata with a component-based, ontology driven framework. In *In Proceedings of K-Cap 2001* (Victoria, BC, Canada, October 2001).

[19] HENDLER, J. Agents and the semantic web. *IEEE Intelligent Systems Journal 16*, 2 (2001), 30–37.

[20] HUMPHREYS, K., GAIZAUSKAS, R., AZZAM, S., HUYCK, C., MITCHELL, B., CUNNINGHAM, H., AND WILKS, Y. Description of the university of sheffield lasie-ii system as used for MUC-7. In *Proceedings of the 7th Message Understanding Conference* (1998). www.itl.nist.gov/iaui/894 02/related_projects/muc/.

[21] KOGUT, P., AND HOLMES, W. Applying information extraction to generate daml annotations from web pages. In *Proceedings of the K-CAP 2001 Workshop Knowledge Markup & Semantic Annotation* (2001). Victoria B.C., Canada.

[22] KUSHMERICK, N., WELD, D., AND DOORENBOS, R. Wrapper induction for information extraction. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1997.* (1997).

[23] MILLER, S., CRYSTAL, M., FOX, H., RAMSHAW, L., SCHWARTZ, R., STONE, R., AND WEISCHEDEL, R. 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_projects/muc/.

[24] MUC7. *Proceedings of the 7th Message Understanding Conference (MUC7).* Nist, 1998. www.itl.nist.gov/iaui/894 02/related_projects/muc/.

[25] MUSLEA, I., MINTON, S., AND KNOBLOCK, C. Wrapper induction for semistructured web-based information sources. In *Proceedings of the Conference on Automated Learning and Discovery (CONALD), 1998.* (1998).

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

[27] VARGAS-VERA, M., MOTTA, E., DOMINGUE, J., LANZONI, M., STUTT, A., AND CIRAVEGNA, F. MnM: Ontology driven semi-automatic or automatic support for semantic markup. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW02* (2002), Springer Verlag.