

promoting access to White Rose research papers



Universities of Leeds, Sheffield and York
<http://eprints.whiterose.ac.uk/>

This is an author produced version of a paper published in **Lecture Notes In Computer Science**

White Rose Research Online URL for this paper:

<http://eprints.whiterose.ac.uk/3951/>

Published paper

Ciravegna, F., Dingli, A., Petrelli, D. and Wilks, Y. (2002) *User-system cooperation in document annotation based on information extraction*. In: Gomez-Perez, A. and Benjamins, R., (eds.) Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 2 - 4 October 2002, Sigüenza, Spain. Lecture Notes In Computer Science (2473). Springer , London, pp. 122-137.

User-System Cooperation in Document Annotation based on Information Extraction

Fabio Ciravegna¹, Alexiei Dingli¹, Daniela Petrelli² and Yorick Wilks¹

¹ Department of Computer Science, University of Sheffield, Regent Court, 211 Portobello Street, S1 4DP, Sheffield, UK, email {fabio|alexiei|yorick}@dcs.shef.ac.uk

² Department of Information Studies, University of Sheffield, Regent Court, 211 Portobello Street, S1 4DP, Sheffield, UK, email D.Petrelli@shef.ac.uk

Abstract. The process of document annotation for the Semantic Web is complex and time consuming, as it requires a great deal of manual annotation. Information extraction from texts (IE) is a technology used by some very recent systems for reducing the burden of annotation. The integration of IE systems in annotation tools is quite a new development and there is still the necessity of thinking the impact of the IE system on the whole annotation process. In this paper we initially discuss a number of requirements for the use of IE as support for annotation. Then we present and discuss a model of interaction that addresses such issues and Melita, an annotation framework that implements a methodology for active annotation for the Semantic Web based on IE. Finally we present an experiment that quantifies the gain in using IE as support to human annotators.

1. Introduction

The effort behind the Semantic Web (SW) is to add information to web documents in order to access knowledge instead of unstructured material, allowing knowledge to be managed in an automatic way. Much effort has been spent in developing methodologies for enriching documents, mainly requiring manual insertion of annotation. It is reasonable to expect users to manually annotate new documents up to a certain degree, but annotation is a slow time-consuming process that involves high costs. Therefore it is vital for the Semantic Web to produce automatic or semi-automatic methods for document enrichment, either to help in annotating new documents or to extract additional information from existing unannotated or partially annotated documents. Information Extraction from texts (IE) can provide the backbone for such tools. IE is an automatic method for locating important facts in electronic documents. In the SW context, IE can be used for document annotation either in an automatic way (via 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 is an area of Natural Language Processing with a long history. Its development has been mainly driven by the MUC conferences, a number of competitive exercises supported by Darpa. One of the main issues in IE is the way in which applications are

defined. The main constraint in the MUC conferences is that applications are to be developed in a short time (e.g. one month). The MUCs represent a scenario in which the cost of new application is not considered important: by bounding the development time they did not put an upper bound neither to the amount of personnel needed for the application nor to the skills used [1]. As a result, most of the systems were portable by IE expert only.

The Semantic Web represents a completely different scenario where the cost is the issue. The rapid and uncontrolled growth of the Web in the last years is mainly due to the simplicity and effectiveness of HTML. Everyone can make available his/her own pages at nearly no cost (the cost of a PC and a telephone line) with very limited skills (i.e. mainly the ability of using a web editor). If we want the Semantic Web to become the widespread evolution of the current Web we have to provide methodologies with the same type of requirement: portability with limited skills and no (or very limited) cost. The requirement is to be extended to all the tools necessary for building the SW. If IE is to be used for annotation, it must be usable at no cost (exactly as web browsers are free) with limited skills. The kind of IE technologies that require experts in IE can be afforded only by big companies and or big service providers (e.g. search engines companies) and can be used for generic indexing. EuroDAML, [2] is an example of a tool that requires an expert to adapt the system to new applications and that is used for very generic IE for the Web (e.g. named entity recognition). The situation is different in scenarios with distributed agents that provide local services. For example a university department wanting to provide a SW service for their Web pages. In this case they will need to define a specific indexing service themselves. The available budget here is very low and the available skills are quite limited (e.g. a student want-to-be web designer and a system manager). No experts in IE can be envisaged here, nor does the budget allow hiring an expensive external company. In an IE perspective for the SW there is the clear need to allow users with no knowledge of IE to build applications (e.g. specialized annotation services for the set of pages).

Adaptive IE systems (IES) use Machine Learning to learn how to adapt to new applications/domains using only annotated corpora [3] 4][5]. They can be adapted to provide annotations for the SW: they monitor the annotations inserted by the user and learn how to reproduce them. When equivalent cases are encountered, annotations are automatically inserted by the IES and users have just to check them. Some new annotation tools for the Semantic Web are starting including adaptive IE as support to annotation. At the Open University, the MnM annotation tool [6] interfaces with both the UMass IE tools¹ and Sheffield's Amilcare². At the University of Karlsruhe the Ontomat annotizer [7] interfaces with Sheffield's Amilcare. The current methodology of interaction between annotation tool and IES is still quite simplistic, influencing also the way in which users and annotation system interacts. Generally a batch interaction mode is adopted, i.e., the user annotates a batch of texts and the IE tool is trained on the whole batch. Then annotation is started on another batch of texts and the IE system proposes annotations to users when cases similar to those found in the training batches are recognized. Although the use of adaptive IE constitutes quite an

¹ www-nlp.cs.umass.edu/software/badger.html

² www.dcs.shef.ac.uk/~fabio/Amilcare.html

improvement with respect to the completely manual annotation approach, in our opinion the tremendous potentialities of adaptive IE technologies are not fully exploited. We believe that it is time to consider the way in which the interaction can be organized in order to both maximize effectiveness in the annotation process and minimize the burden of annotating/correcting on the user's side. We expect that such change will also influence the user-annotation tool interaction style by moving from a simplistic user-system interaction to real user-system collaboration³. We propose two user-centered criteria as measure of appropriateness of this collaboration: *timeliness* and *intrusiveness* of the IE process. 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.

Timeliness: when the IE system (IES) is trained on blocks of texts, there is a time gap between the moment in which annotations are inserted by the user and the moment in which they are used by the system for learning. User and system work in strict sequence, one after the other. This sequential scheduling hampers true collaboration. If a batch of texts contains many similar documents, users may spend considerable amount of time in annotating similar documents without receiving feedback from the IES for the simple reason that no learning is scheduled for the moment. The IES is not supportive to the user neither the user effort is very useful, since similar cases are of very little use for the learner because they cannot offer the variety of phenomena that empower learning. The bigger the size of the batch of texts the worse, the problem of lack of timeliness is. A true collaboration implies a (re)training of the system after every annotated text is released by the user. Training can take a considerable amount of CPU time, therefore stop the annotation session for a while. A positive collaboration requires not to constraint the user time to the IES training time (otherwise intrusiveness increases). We believe that an intelligent scheduling is needed to keep timeliness in learning without increasing intrusiveness.

Intrusiveness: the IE system can bother users in a number of ways, for example by proposing annotations generated by unreliable rules (e.g. induced using an insufficient number of cases). A positive collaboration requires to enable users to tune the proactivity of the IE system in order to avoid intrusiveness.

In this paper we present an IE-based annotation methodology for the Semantic Web that takes into account the problems of timeliness and intrusiveness mentioned above. Moreover we quantitatively evaluate the support provided by IE in a simulation of experiment of text annotation.

³ Collaboration means working together for a common goal, all partners contributing with their own capabilities and skills.

2. Towards a new interaction model

We propose an interaction model that aims at producing a non-intrusive and timely support for users during the annotation process. In this section we describe the way in which user and system interact and discuss how such requirements are met by our model.

2.1. User-system interaction

We split the annotation process 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 IES.

During **training** users annotate texts without any contribution from the IES. Here the IES uses the user annotations to train its learner. During this phase the IES is constantly inducing rules. We can define two sub-phases: (a) *bootstrapping* and (b) *training with verification*. During bootstrapping the only IES task is to learn from the user annotations. This sub-phase can be of different length, depending on the minimum number of examples needed for a minimum of training. During the second sub-phases, the user continues with the unassisted annotation, but the IES behaviour changes, as it uses its induced rules to silently compete with the user in annotating the document. 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 learners. The training phase ends when the IES accuracy reaches the user preferred level of pro-activity. 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 the 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 the user in annotating; the user feeds back the mistakes

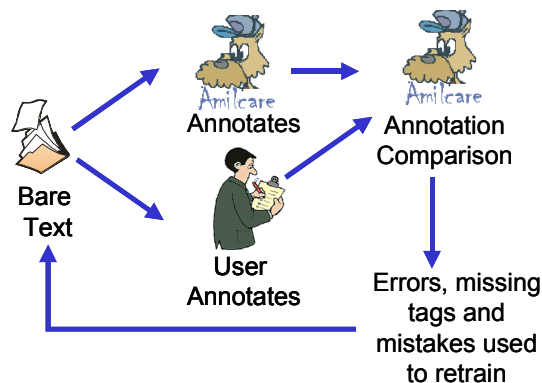


Figure 1. The training with verification sub-phase. In this figure Amilcare is used as example of adaptive IES.

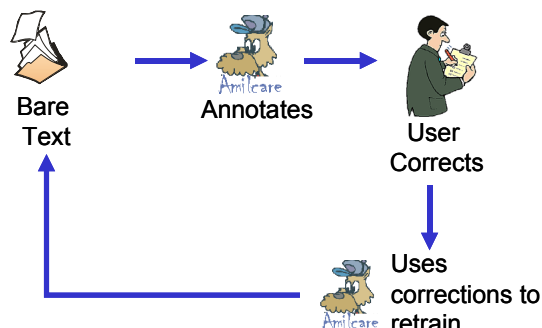


Figure 2. The active annotation with revision phase

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.

2.2. Coping with Intrusiveness

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 annotation interface or any specific adaptation by the user. It integrates in the usual user environment and provides suggestions in a way that is both familiar and intuitive for the user. To some extent users could even ignore that the IES is working for them.

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. Some IES provide internal tuning methods for balancing features such as precision and recall or the minimum number of cases to be covered in order to accepted a rule for annotation. Such tuning methodologies are designed for IE experts since they require a deep knowledge of the underlying IE system. This is especially true because the user's goal is tuning the level of intrusiveness in the annotation process and very often there is no obvious correspondent in the IES tuning methodology. For example Amilcare allows to modify error thresholds for rules, number of cases covered by rules for acceptance, balance of precision and recall in rule tuning: none of these correspond directly to tuning the level of intrusiveness (even if large part of it relies in the precision/recall balance). Moreover, 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. A user-friendly interaction methodology requires enabling the user in selecting the appropriate level of intrusiveness, without coping with the complexity of tuning an adaptive IE system. In our model the annotation interface 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 [8]. This is important

because the annotation interface is a tool designed for specific user classes and therefore able to elicit tuning requirements by using 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). As explained above, the batch approach presents timeliness problems. We propose background learning to provide timely support without intrusiveness. If we observe how time is spent in the annotation process (select a document, manually annotate the document, save the annotation), we notice that most of the user time is spent in the manual annotation process. This is the right moment to train the IES in the background without the user noticing it. In principle it is possible to treat every annotation event in the interface as a request to train on a specific example, but this requires the ability to retreat annotations in case of user errors, making the interaction with the IES quite complex. In our approach 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 extract information from document n (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 the user, as the annotation of a document generally requires a great deal more time than training on a single text.

2.3. Coping with Timeliness

Timeliness means just in time learning from previous user annotations. 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. Considering that in many applications the order in which documents are annotated is unimportant, in such cases it is possible to organize the annotation order so to avoid the possibility of presenting similar documents in sequence and therefore to hide the small lack of timeliness. In order to implement such feature we need a measure of similarity of texts from the annotation point of view. The IES can be used to work out such a measure. At the end of each learning session all the induced rules are applied to the unannotated part of the corpus so to identify two main subsets: texts where the available rules fire (i.e. annotations can be added: positive subset) and texts where they do not fire at all (uncovered texts: negative subset). Each text in the positive subset can be associated with a score given by the number of annotations that can be added. The score can be used as an approximation of similarity among texts: inserted annotations mean similarity with

respect to the part of the corpus annotated so far, no inserted annotation means actual difference. Such information can be used to make the timeliness more effective: a completely uncovered document is always followed by a fairly covered document. In this way a difference between successive documents is very likely and therefore the probability that similar documents are presented in turn within the batch of two (i.e. the blindness window of the system) is very low. Incidentally this strategy also tackles another major problem in annotation, i.e. user boredom, which can make the 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.

In the next section a first implementation of the presented interaction model is presented. We introduce both the IES used (Amilcare) and the annotation interface (Melita). Finally we discuss how the current implementation meets the requirements described.

3. Adaptive IE in Amilcare

The model above requires an adaptive IES to strictly cooperate with the user. In our implementation we have used Amilcare⁴. Amilcare is a tool for adaptive Information Extraction from text (IE) designed for supporting active annotation of documents for the Semantic Web. In its standard version it performs IE by enriching texts with XML annotations, i.e. the system marks the extracted information with XML annotations. In the Semantic Web version in which it is supposed to be interacting with an annotation tool, it actually leaves the text unchanged and it returns the extracted information as a triple <annotation, startPosition, endPosition> so to let the annotation tool decide how to actually annotate the text. The only knowledge required for porting Amilcare to new applications or domains is the ability of manually annotating the information to be extracted in a training corpus. No knowledge of IE is necessary.

Adaptation starts with the definition of a tag-set for annotation possibly organized as an ontology where tags are associated to concepts and relations. Then users have to manually annotate a corpus for training the learner. An annotation interface is to be connected to Amilcare for annotating texts, e.g. using XML-based mark ups. As mentioned Amilcare has been integrated with a number of annotation tools so far, including MnM[6], Ontomat[7]. For example MnM automatically converts the user annotations into XML tags to train the learner. Amilcare's learner induces rules that are able to reproduce such annotation. Amilcare can work in two modes: training, used to adapt to a new application, and extraction, used to actually annotate texts. In both modes, Amilcare first of all preprocesses texts using Annie, the shallow IE system included in the Gate package ([9], www.gate.ac.uk). Annie performs text tokenization (segmenting texts into words), sentence splitting (identifying sentences) part of speech tagging (lexical disambiguation), gazetteer lookup (dictionary lookup) and Named Entity Recognition (e.g. proper names spotting and classification).

⁴ www.dcs.shef.ac.uk/~fabio/Amilcare.html

When operating in training mode, Amilcare induces rules for information extraction. The learner is based on (LP)², a covering algorithm for supervised learning of IE rules based on Lazy-NLP [10] [11]. This is a wrapper induction methodology [12] that, unlike other wrapper induction approaches, uses linguistic information for rule generalization. The learner starts inducing wrapper-like rules that make no use of linguistic information, where rules are sets of conjunctive conditions on adjacent words. Then the linguistic information provided by Annie is as the basis for rule generalization: conditions on words are substituted with conditions on the linguistic information (e.g. condition matching either the lexical category, or the class provided by the gazetteer, etc. [11]). All the generalizations are tested in parallel and the best k generalizations are kept for IE. The idea is that the linguistic-based generalization is used only when the use of NLP information is reliable or effective. 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. Lazy NLP-based learners learn which is the best strategy for each information/context separately. For example they may decide that using the result of a part of speech tagger is the best strategy for recognizing the speaker in seminar announcements, but not to spot the seminar location. This strategy is quite effective for analysing documents with mixed genres, quite a common situation in web documents [13].

The learner induces two types of rules: tagging rules and correction rules. A tagging rule is composed of a left hand side, containing a pattern of conditions on a connected sequence of words, and a right hand side that is an action inserting an XML tag in the texts⁵. Correction rules correct imprecision, i.e. shift misplaced tags to the correct position. They are learnt from the mistakes made in attempting to re-annotate the training corpus using the induced tagging rules. The output of the training phase is a collection of rules for IE that is associated to the specific scenario. When working in extraction mode, Amilcare receives as input a (collection of) text(s) with the associated scenario (including the rules induced during the training phase). It preprocesses the text(s) by using Annie and then it applies its rules and returns the original text with the added annotations (or just the annotation triples in the SW version).

With Amilcare it is possible to define automatic or semiautomatic services for the SW with limited skills (the ability of annotating the texts) and limited cost (the number of texts to be annotated for training –as we will see- is quite limited). For example the university department mentioned in the introduction could use the student creating the pages to annotate the pages. Amilcare would learn in the background without requiring any specific adaptation except the definition of the annotation set (necessary in any case for defining SW services). This is the reason why some annotation tools include Amilcare as support to annotation.

⁵ In the SW version no tag is actually inserted in the text; as mentioned a triple *<annotation, startIndex, endPosition>* is returned to the external annotation interface.

4. The Melita framework

Melita is an ontology-based demonstrator for text annotation. The goal of Melita is not to produce a further annotation interface, but a demonstrator of how it is possible to actively interact with the IES in order to meet the requirements of timeliness and tuneable pro-activity mentioned above. Melita's main control panel is depicted in figure 3. It is composed of two main areas:

1. The ontology (left) representing the annotations that can be inserted; annotations are associated to concepts and relations. A specific color is associated to each node in the ontology (e.g. "speaker is depicted in blue).
2. The document to be annotated (center-right). Selecting the portion of text with the mouse and then clicking on the node in the ontology insert annotations. Inserted annotations are shown by turning the background of the annotated text portion to the color associated to the node in the hierarchy (e.g. the background of the portion of text representing a speaker becomes blue).

Melita does not differ in appearance from other annotation interfaces such as the Gate annotation tool, or MnM or Ontomat. This is because – as mentioned – it is a demonstrator to show how a typical annotation interface could interact with the IES. The novelty of Melita is the possibility of (1) tuning the IES so to provide the desired level of proactivity and (2) scheduling texts so to provide timeliness in annotation learning. The typical annotation cycle in Melita follows the two-phase cycle based on training and active annotation described in the previous section. Users may not be aware of the difference between the two phases. They just will notice that at some point the annotation system will start suggesting annotations and that they have a way to influence when and with which modalities this will happen.

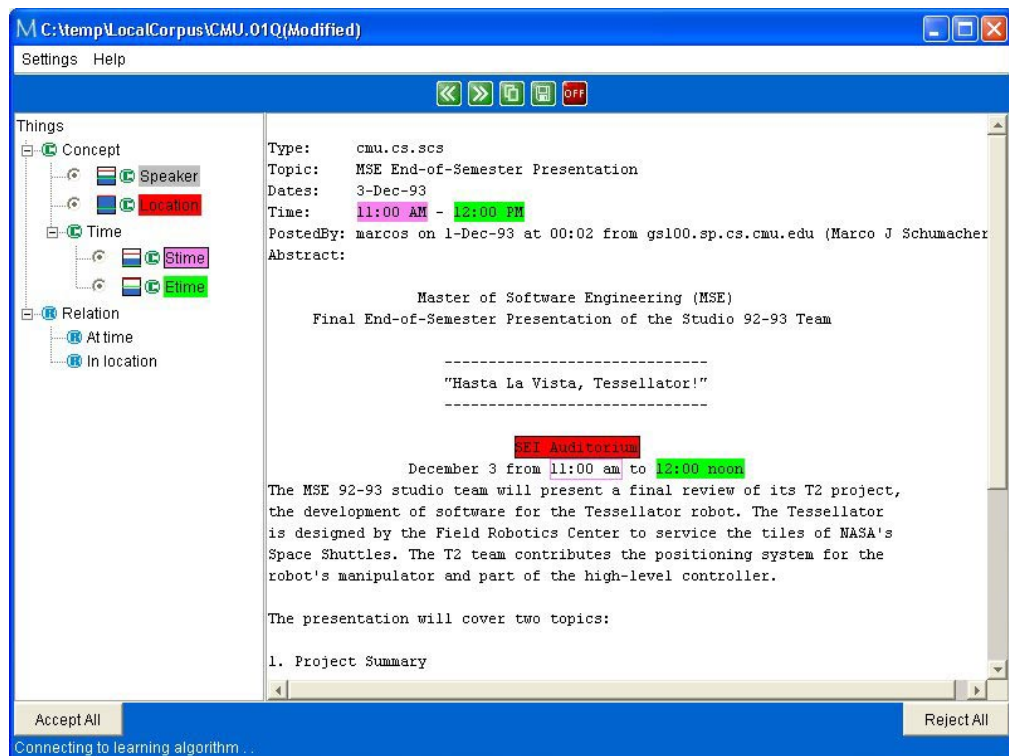


Figure 3: The Melita annotation Interface

4.1. Suggesting Annotations

There are two ways in which Melita can suggest annotations to users, according to the reliability of such suggestions. For suggestions Amilcare is quite sure about, Melita will present them in the document panel in a way similar to the annotations inserted by the user. The background of the text where the information has been found turns into the specific annotation colour (e.g. grey for speaker in figure 3). The difference with respect to the actual user annotations is that a darker border surrounds them in order to be easily spotted for user checking. For example in figure 3 the location “SEI Auditorium” highlighted in red is a reliable Amilcare’s suggestion, while “12 PM” is a user defined annotation. In case of suggestions Amilcare is less sure about, they are presented in a different way. The background is left unchanged (white), but a coloured border (the same colour of the potential annotation, e.g. grey for speaker) surrounds the text. For example “11 am” (at the text centre in figure 3) is a suggestion of this type. They are easy to spot by the user, but they are marked as unreliable. A difference in the suggestion’s semantics corresponds to the difference in presentation: reliable annotations are supposed to be correct; a user action is required to remove them if they are wrong. Less reliable annotations are supposed to be just suggestions to the user; an action is required to confirm them; otherwise they will not be saved with the text in the end. We believe that both annotation types are useful as they allow to clearly communicating the user what suggestions are to be trusted and which are just a reasonable guess. Reasonable guesses are presented for two reasons: first of all they represent a situation in which the learner requires user feedback: removing such information means a clear message to the learner that the guess is wrong and therefore rules are to be changed. From the user point of view guesses are very often useful because they are often imprecise but nonetheless they tend to correctly identify the area in which such information is present even if the information is not correctly identified (e.g. in “at <time> 3:00</time> pm” the annotation is imprecise – pm should be part of the time – but it is useful to focus the user attention on the place where the correct annotation should go). Note that reliability can vary for different pieces of information. For example a system can become quite reliable in a short time in recognizing some information (e.g. seminar start time) requiring more training examples for others (e.g. speaker). In this case there will be a moment in which the suggested annotations for the time will be reliably inserted (i.e. with coloured background) while the annotations for the speaker will be less reliable (presented with coloured border only).

4.2. Balancing Proactivity

Users must be empowered to customize the strategy above, participating in the definition of what is reliable information and what is not. Also some very unreliable suggestions can be not presented, and – again – we want to empower the user to say which of them are not to be presented. This means that users must be empowered to control proactivity (and therefore intrusivity). In Melita, users can customize the behaviour of the IES, i.e. tuning the IES’s level of proactivity, by using a special sidebar (fig.4). It allows to set two thresholds that divide the accuracy space in three

areas: the first level decides which is the minimum accuracy the IES must be able to reach in order to start considering annotations as reliable. The second threshold defines the minimum accuracy the system must reach before starting presenting less reliable suggestions. In the example in figure 4 the system will consider reliable (and therefore suggest with coloured background) when the annotation accuracy is greater than 75%. Annotations that do not reach 75% reliability are still suggested (using the coloured border only) if they reach at least 43% of reliability. When accuracy is less than 43% the IES does not suggest at all. There is a general default that can be customised and holds for all the nodes in the ontology and that can be overridden for specific nodes by using the same kind of window. Changing the default for specific annotations (e.g. “speaker”) is useful because users can have different feelings about

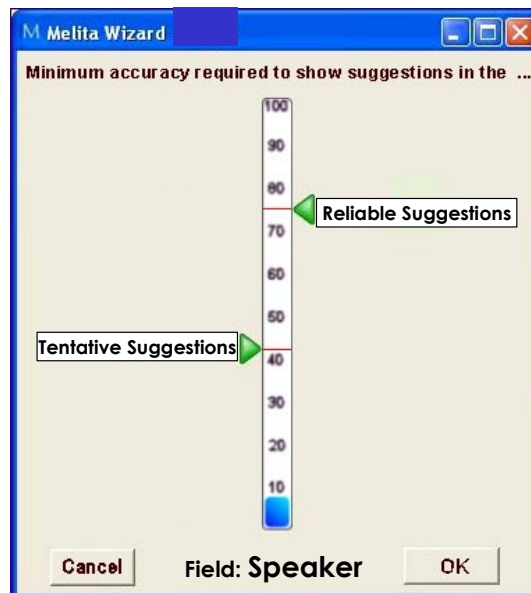


Figure 4. the slider to customize intrusiveness

intrusiveness for different kinds of information. Note that users do not need to know in detail what 45% means. They can easily reason from a qualitative point given the current IES behaviour. If the user feels that the IES is not proactive enough, s/he can decide to lower (one of the) two thresholds. If the system is intrusive the user can decide to raise them. For turning off all the system suggestions it is just necessary to raise both the thresholds above 100%. Moreover the more you move in either direction, the more the effect on the IES will be relevant. It is important that the thresholds are independent because users can have different feeling on intrusiveness for the different suggestion modes. The same slider shows also the average accuracy currently reached by the IES in annotating a specific information type: a blue filler mark grows from the bottom (around 10% in figure 4). It represents the distribution of accuracy of the potential suggestions for the specific annotation. Such

information can be used in tuning proactivity: less intrusivity=raise a threshold above the average, more proactivity, move a threshold below the average.

5. An Experiment on IE's Effectiveness

We performed a number of experiments for demonstrating how fast the IES converges to an active annotation status and to quantify its contribution to the annotation task, i.e. its ability to suggest correctly. We selected a subset of the Computer Science Jobs announcement corpus, manually annotated by M. E. Califf [14]. This is a corpus used for evaluating adaptive IE algorithms on semi-structured texts [15]. The subtask we selected was to recognise in a set of 250 news posts about job offers for computer scientists: the city, country and state in which the job is offered, the company offering the job, the actual recruiter, the required knowledge about both computer languages and platforms, and the offered salary. We believe that this task can be considered a representative task for the Semantic Web.

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 accuracy of the IES on the following 100 texts in the corpus (so when training on 25 texts, the test was performed also on the following 25 texts to 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. Results are shown in the figure at the end of the paper. On the X-axis the number of documents provided for training is shown. On the Y-axis precision, recall and f-measure⁶ are presented.

The maximum support comes in annotating city, country, state and posting date. This is not surprising as they present quite regular fillers. Other experiments on other corpora have shown that an equivalent gain can be obtained also for annotations requiring time expressions as fillers. After training on only 10 texts, the system is potentially able to propose 253 instances of cities (out of 303 present in the corpus), 228 are correct, 22 are wrong, 3 partially correct⁷, 72 missing, leading to Precision=90 Recall=75 (see figure 5 and table 1). This is possible because of Amilcare's ability to generalize over both the text context and the gazetteer information provided by Annie, where a list of locations is present. Please note that the recognition of cities, state and country is not a simple Named Entity Recognition task. The system must not only recognise the name of a place, but also recognise that such place is the location of work. There are other locations in the texts that are irrelevant (e.g. in the address of the recruiter) and only the job location must be recognised. This implies the ability to recognise the context in which the location name appears. The same applies to the posting date: there are many other dates in the texts and only the correct one must be

⁶ A balanced average of precision and recall.

⁷ Where the proposed and correct annotations partially overlap. They count as half correct in calculating precision and recall.

identified. The situation is more complex for other fields such as recruiter or company, where 80% F-measure is reached after 100 texts. These annotations are much more difficult to learn than expressions whose filler are either very regular (e.g. time or date expressions) or can be listed in a gazetteer (we did not have a suitable list of companies), because their regularity is much less direct. We performed the same type of analysis on other corpora for adaptive IE, the CMU seminar announcements corpus, where 483 emails are manually annotated with speaker, starting time, ending time and location of seminars (www.isi.edu/~muslea/RISE/) and found analogous results.

Table 1. Amount of training texts needed for reaching at least 75% precision and 50% recall

Tag	Amount of Texts needed for training	Prec	Rec	F-measure
City	10	90	75	82
country	10	81	92	86
state	5	79	87	83
company	100	91	72	86
recruiter	30	81	50	62
language	50	80	59	68
platform	50	77	52	62
salary	5	75	54	62
post_date	5	97	100	98

The above 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 job announcement task, our experiments show that it is possible to move from bootstrapping to active annotation after annotating a very limited amount of texts. In table 1 we show the amount of training needed for moving to active annotation for each type of information, given a minimum user requirement of 75% precision. 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., known location names). In user terms this means that it is possible to focus the activity on annotating more complex pieces of information (e.g. company and recruiter), avoiding to be bothered with easy and repetitive ones (such as locations). With some more training cases the IES is also able to contribute in annotating the complex cases. onclusions and future work

IES can strongly support users in the annotation task, alleviating users from a big deal of the annotation burden. Our experiments show that such help is particular strong and immediate for repetitive or regular cases, allowing focusing the expensive and time-consuming user activity on more complex cases. In our experiment we have quantified such support for an experiment about job announcements. Despite these positive results, we claim that the simple quantitative support is not enough. An interaction methodology between annotation interface, user and IES is necessary in order to reduce intrusivity and maintain timeliness of support. The methodology proposed in this paper addresses such concern, as:

1. It inserts in the usual user environment without imposing particular requirements on the annotation interface used to train the IES (reduced intrusiveness).
2. It maximizes the cooperation between user and IES: users insert annotations in texts as part of their normal work and at the same time they train the IES. The IES in turn simplifies the user work by inserting annotations similar to those inserted by the user in other documents; this collaboration is made timely and effective by the fact that the IES is retrained after each document annotation.
3. The modality in which the IES system suggests new annotations is fully tuneable and therefore easily adaptable to the specific user needs/preferences (intrusiveness is taken under control).
4. It allows to timely train the IES without disrupting the user pace with learning sessions consuming a large amount of CPU time (and therefore either stopping or slowing down the annotation process).

There are two open issues that arise from our experience. On the one hand the effect on the user of excellent IES performances after a small amount of annotation is still to be considered. For example when $P=90$, $R=75$ is reached after only 10 texts (as for company in the jobs announcement task), users could be tempted to rely on the IES suggestions only, avoiding any further action apart from correction. This would be bad not only for the quality of document annotation, but also for the IES effectiveness. As a matter of fact, each new annotated document is used for further training. Rules are developed using existing annotations. They are tested on the whole corpus to check against false positives (e.g. the rest of the corpus is considered a set of negative examples). A corpus with a relevant number of missing annotations provides a relevant number of (false) negative examples that disorients the learner, degrading its effectiveness and therefore producing worse future annotation. The entire dimension of the problem is still to be analysed. We are currently considering applying strategies such as randomly removing annotations in order to test the user attention. On the other hand the time saved by using an IES is still to be quantified. The experiments above seem to suggest a strong reduction of annotation time, but we intend to actually measure the improvement in experiments with real users.

Acknowledgements

This work was carried out within the AKT project (<http://www.aktors.org>), sponsored by the UK Engineering and Physical Sciences Research Council (grant GR/N15764/01). AKT involves the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and the Open University. Its objectives are to develop technologies to cope with the six main challenges of knowledge management: acquisition, modelling, retrieval/extraction, reuse, publication and maintenance. Thanks to Enrico Motta, Mattia Lanzoni, John Domingue, Steffen Staab and Siegfried Handschuh for a number of useful discussions. Thanks to the Gate group for providing Annie (www.gate.ac.uk) and for help in integrating it into Amilcare.

Bibliography

1. F. Ciravegna, A. Lavelli, G. Satta: 'Bringing information extraction out of the labs: the Pinocchio Environment', in *ECAI2000, Proc. of the 14th European Conference on Artificial Intelligence*, ed., W. Horn, Amsterdam, 2000. IOS Press

2. P. Kogut and W. Holmes: "Applying Information Extraction to Generate DAML Annotations from Web Pages", K-CAP 2001 Workshop Knowledge Markup & Semantic Annotation, Victoria B.C., Canada (2001).
3. M. E. Califf, D. Freitag, N. Kushmerick and I. Muslea (eds.): AAAI-99 Workshop on Machine Learning for Information Extraction, Orlando Florida (1999), <http://www.isi.edu/~muslea/RISE/ML4IE/>
4. R. Basili, F. Ciravegna, R. Gaizauskas (eds.) ECAI2000 Workshop on Machine Learning for IE, Berlin (2000), www.dcs.shef.ac.uk/~fabio/ecai-workshop.html
5. F. Ciravegna, N. Kushmerick, R. Mooney and I. Muslea (eds.), IJCAI-2001 Workshop on Adaptive Text Extraction and Mining held in conjunction with the 17th International Conference on Artificial Intelligence, Seattle, (2001), <http://www.smi.ucd.ie/ATEM2001/>
6. 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", Proc. of the 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW02, Sigüenza, Spain (2002).
7. S. Handschuh, S. Staab and F. Ciravegna: "S-CREAM - Semi-automatic CREATION of Metadata", Proc. of the 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW02, Sigüenza, Spain, (2002).
8. F. Ciravegna and D. Petrelli: "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 Conference on Artificial Intelligence, Seattle (2001).
9. D. Maynard, V. Tablan, H. Cunningham, C. Ursu, H. Saggion, K. Bontcheva and Y. Wilks: "Architectural Elements of Language Engineering Robustness", Journal of Natural Language Engineering, Special Issue on Robust Methods in Analysis of Natural Language Data, forthcoming in 2002.
10. F. Ciravegna: "Adaptive Information Extraction from Text by Rule Induction and Generalisation" in Proceedings of 17th International Joint Conference on Artificial Intelligence (2001).
11. F. 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 Conference on Artificial Intelligence (IJCAI-01), Seattle, August, 2001
12. N. Kushmerick, D. Weld and R. Doorenbos: 'Wrapper induction for information extraction', Proc. of 15th International Conference on Artificial Intelligence, Japan (1997).
13. F. Ciravegna: "Challenges in Information Extraction from Text for Knowledge Management", IEEE Intelligent Systems and Their Applications, 16-6, November, (2001).
14. M. E. Califf: 'Relational Learning Techniques for Natural Language' IE, *Ph.D. thesis*, Univ. Texas, Austin, (1998), www.cs.utexas.edu/users/mecaliff
15. D. Freitag and N. Kushmerick, 'Boosted wrapper induction', in R. Basili, F. Ciravegna, R. Gaizauskas (eds.) *ECAI2000 Workshop on Machine Learning for Information Extraction*, Berlin, 2000, www.dcs.shef.ac.uk/~fabio/ecai-workshop.html.

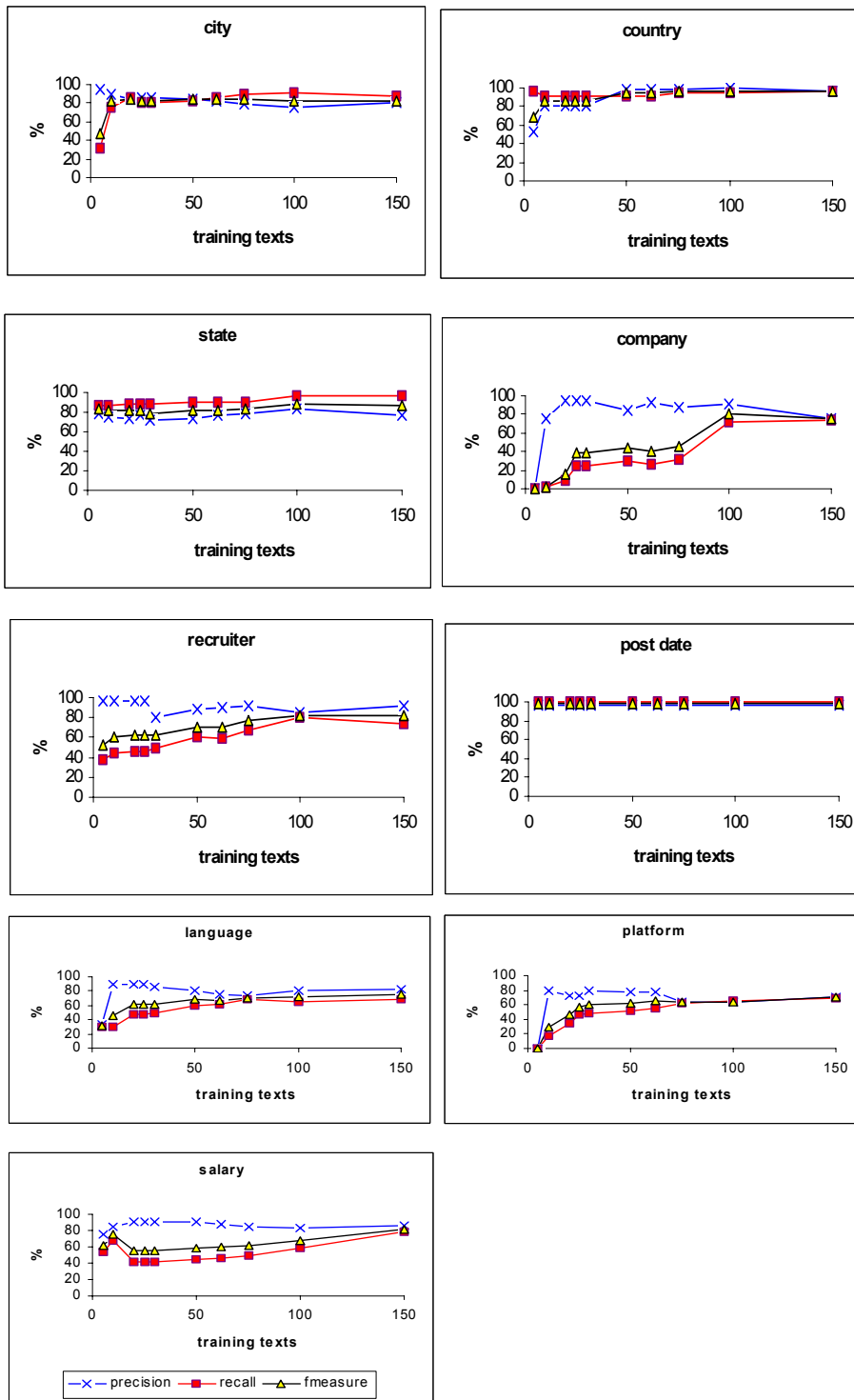


Figure 5. The learning curve for the different information in the job task