Timely and Non-Intrusive Active Document Annotation via Adaptive Information Extraction

Fabio Ciravegna[1], Alexiei Dingli[1], Daniela Petrelli[2] and Yorick Wilks[1]

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 of the most recent systems for actively supporting users in the process and reducing the burden of annotation. The integration of IE systems in annotation tools is quite a new development and in our opinion there is still the necessity of thinking the impact of the IE system in the process of annotation. In this paper we discuss two main requirements for active annotation: timeliness and tuning of intrusiveness. Then we present and discuss a model of interaction that addresses the two issues and Melita, an annotation framework that implements such methodology.

1. INTRODUCTION

The effort behind the Semantic Web (SW) is to add content to web documents in order to access knowledge instead of unstructured material, allowing knowledge to be managed in an automatic way. Much work is done on (1) the definition of standards for representation of knowledge (e.g. XML, RDF, OIL), (2) the definition of structures for knowledge organization (e.g. ontologies) and (3) the population of such knowledge structures. (1) and (2) actually provide the necessary infrastructure for the Semantic Web. (3) actually requires methodologies for creating semantically enriched documents. 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 extracting information from web-related documents, either for helping in annotating new documents or to extract additional information from existing unstructured or partially structured documents. In this context, Information Extraction from texts (IE) is one of the most promising areas of Human Language Technologies for the Semantic Web. IE is an automatic method for locating important facts in electronic documents for successive use, e.g. for annotating documents or for information storing (such as populating an ontology with instances). In this perspective IE is the perfect support for knowledge identification and extraction from Web documents as it can – for example - provide support in documents 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). In the last years a big effort has been spent in the IE community on the use of Machine Learning for helping in porting IE systems to new applications/domains [1][2][3]. Some new annotation tools for the Semantic Web already include adaptive IE capabilities for helping in the annotation process. At the Open University, the MnM annotation tool [4] interfaces with both the UMass IE tools [5] and Sheffield's Amilcare [11]. At the University of Karlsruhe the Ontomat annotizer [6], an implementation of the CREAM environment, interfaces with Sheffield's Amilcare. The advantage of using adaptive IE as a support for annotation is quite clear: the IE system monitors the annotations inserted by the user and it learns how to reproduce them. When equivalent cases are encountered, annotations are automatically inserted by the IE system and users have just to check them. This approach, called active learning, has been proven to reduce the burden of manual annotation up to 80% in some cases [7]. The current methodology of interaction between annotation tool and IE system is still quite simplistic. This influences also the way in which the user and the 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 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 collaboration[1]. We propose two user-centred 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 (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

[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

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

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 a lot 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 it is efficient 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 the intrusiveness of the IES increases). We believe that an intelligent scheduling is needed to keep timeliness in learning without increasing intrusiveness. It is also important to bear in mind that timeliness is a matter of perception from the user side, not an absolute feature, therefore what is important is that users do not perceive any delay or impediment. The focus is on effective collaboration not on timeliness at any cost.

Intrusiveness: in all the experiments with active learning done so far it turned out difficult to avoid bothering users with proposed annotations generated by unreliable rules (e.g. induced using an insufficient number of cases). This problem is mainly related to the tuning of the IES behaviour. 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 underline IE system. This is especially true because the user 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). 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. We think that a user-friendly interaction methodology must be implemented to help in selecting the appropriate level of intrusiveness, without requiring users to cope with the complexity of tuning an adaptive IE system.

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.

2. THE ANNOTATION PROCESS

In our model the annotation process is split into two main phases from the system point of view: (1) training and (2) active annotation with revision. In user terms the first corresponds to unassisted annotation, while the latter just requires correction of annotation proposed by the IES.

2.1 Training

During training users annotate texts without any contribution from the IES. In this phase the IES uses the annotations inserted by the user 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 according to the specific IES requirements. It depends on the minimum number of examples needed for a minimum training. During the second sub-phases, the user continues with the unassisted annotation, but the behaviour of the IES changes. With some rules already available the IES silently competes with the user in annotating the document. When the annotation process is finished, 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 accuracy in annotating can provide the user preferred level of pro-activity and therefore it is possible to move to the next phase: active annotation. We will discuss in the following section how this condition is verified.
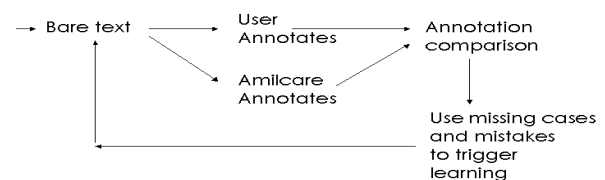

Figure 1. The training with verification sub-phase.

2.2 Active Annotation with Revision

In this phase the annotation methodology is heavily based on the suggestions of the IES and the user main task is to correct and integrate the suggested annotations (i.e. remove or add annotations). Human corrections and integrations 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 annotation; the user feeds back the 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 the user has to insert. 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.

3. A NEW MODEL OF INTERACTION

The proposed model of interaction is based on non-intrusive and timely active annotation. The first level of non-intrusiveness is that the IES does not require any specific interface for annotation or any specific adaptation by the user. It integrates in the usual user environment and provides suggestions for possible annotations in a way that is both familiar and intuitive for the user. To some extent users could even ignore that an IES is working for them. The interaction with the user is left to the annotation interface, a tool designed for specific user classes and therefore

able to elicit the tuning requirements by using the correct terminology for the specific domain. Even the correct settings and requirements for the appropriate IES's settings must be elicited through the interface (and then converted in the IES specific settings thorough an API).

3.1 Intrusiveness vs. Proactivity

Intrusiveness is the risk related to proactivity. As mentioned, there are a number of ways in which the IES can be intrusive with respect to the user task. On the one hand when the system suggests annotation during phase 2 (active annotation with revision), it 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. The annotation interface must bridge 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) [8]. On the other hand the IES training requires CPU time and this can slow down or even stop the user activity. This may happen in both the phases mentioned above (training and active annotation with revision) as discussed in the next section.
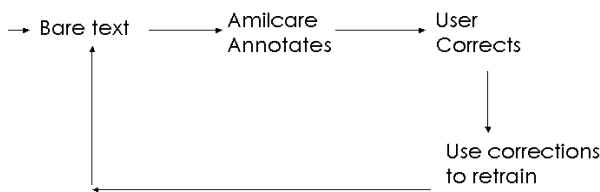

Figure 2. The active annotation with revision phase

3.2 Limiting the User Idle Time

Training requires time and for this reason most of the current systems use a batch mode of training so to limit the time in which the user has to wait while the system trains to specific moments (e.g. coffee time). As explained above, the batch approach presents timeliness problems: users may have to annotate a number of similar texts before the learner is activated and the IES is able to suggest annotations.

An appropriate scheduling of the learning phase can both improve timeliness between user's annotation and system learning and limits the user idle time to the minimum. 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. For this reason we believe that this is the right moment to train the IES in the background without the user noticing it. In principle it would be 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 and this makes the interaction with the IES quite complex. In our method the IES

works in the background with two parallel and asynchronous processes. On the one hand while the user annotates document n the system learns the annotations inserted in document $n-1$, i.e. the learner is always one document behind the user. 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 $n-2$) in order to extract information (either for suggesting annotations during active annotation or in order to silently test its accuracy during unassisted learning). This means that the annotation capability is always two steps behind. 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.

3.3 Coping with Timeliness

As explained above 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, with all the disadvantages of batch training mentioned above (even if a batch of size two is quite small). In this respect the collaboration between the system and the user fails in being effective. Timeliness is a matter of perception from the user side, not an absolute feature, therefore the only important matter – we believe – is that users perceive it. In this respect we start from the consideration that in many applications the order in which documents are annotated is random. Generally users adopt criteria such as date of creation or file name order in directories. 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 lack of timeliness. In order to implement such a 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 whole unannotated corpus. As result two main subsets in the corpus are detected: texts were the available rules fire (i.e. annotations can be added: positive subset) and texts were 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. This is the major reason why the level of user productivity and effectiveness falls proportional to time. Presenting users with radically different documents should avoid the boredom that comes from coping with very similar documents in sequence. In the next section a first implementation of the discussed 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.

4. ADAPTIVE IE IN AMILCARE

Amilcare is a tool for adaptive Information Extraction from text (IE) designed for supporting active annotation of documents for the Semantic Web. It performs IE by enriching texts with XML annotations, i.e. the system marks the extracted information with XML annotations. 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 Human Language Technology 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 using XML mark ups. As mentioned Amilcare has been integrated with a number of annotation tools so far, including MnM[4], Ontomat[6]. For example the annotation interface in Ontomat is used to annotate texts in a user-friendly manner. Ontomat automatically converts the user annotations into XML tags to train the learner. Amilcare's learner induces rules that are able to reproduce the text 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 (recognition of people and organization names, dates, etc.).

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 in the rule generalization process. 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 used in order to generalise rules: 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 by using a variant of the AQ algorithm [13] 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 analyzing documents with mixed genres, quite a common situation in web documents [14].

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 texts. Each rule inserts a single XML tag, e.g. `</speaker>`. This makes the approach different from many adaptive IE algorithms, whose rules recognize whole pieces of information (i.e. they insert both `<speaker>` and `</speaker>`[7]), or even multi slots [15]. Correction rules shift misplaced annotations (inserted by tagging rules) to the correct position. They are learnt from the mistakes made in attempting to re-annotate the training corpus using the induced tagging rules. 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 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. The Gate annotation schema is used for annotation [9].

5. 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 tunable pro-activity mentioned above. Melita's main control panel is depicted in figure 3. It is composed of three main areas:

1. The *ontology* (left) representing the annotations that can be inserted; annotations are associated to concepts and relations. A specific colour is associated to each node in the ontology (e.g. "speaker is depicted in blue).

2. The *document to be annotated* (centre-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 colour associated to the node in the hierarchy (e.g. the background of the portion of text representing a speaker becomes blue).

3. The *IES suggestion area* (bottom) where some of the suggested annotations are presented.
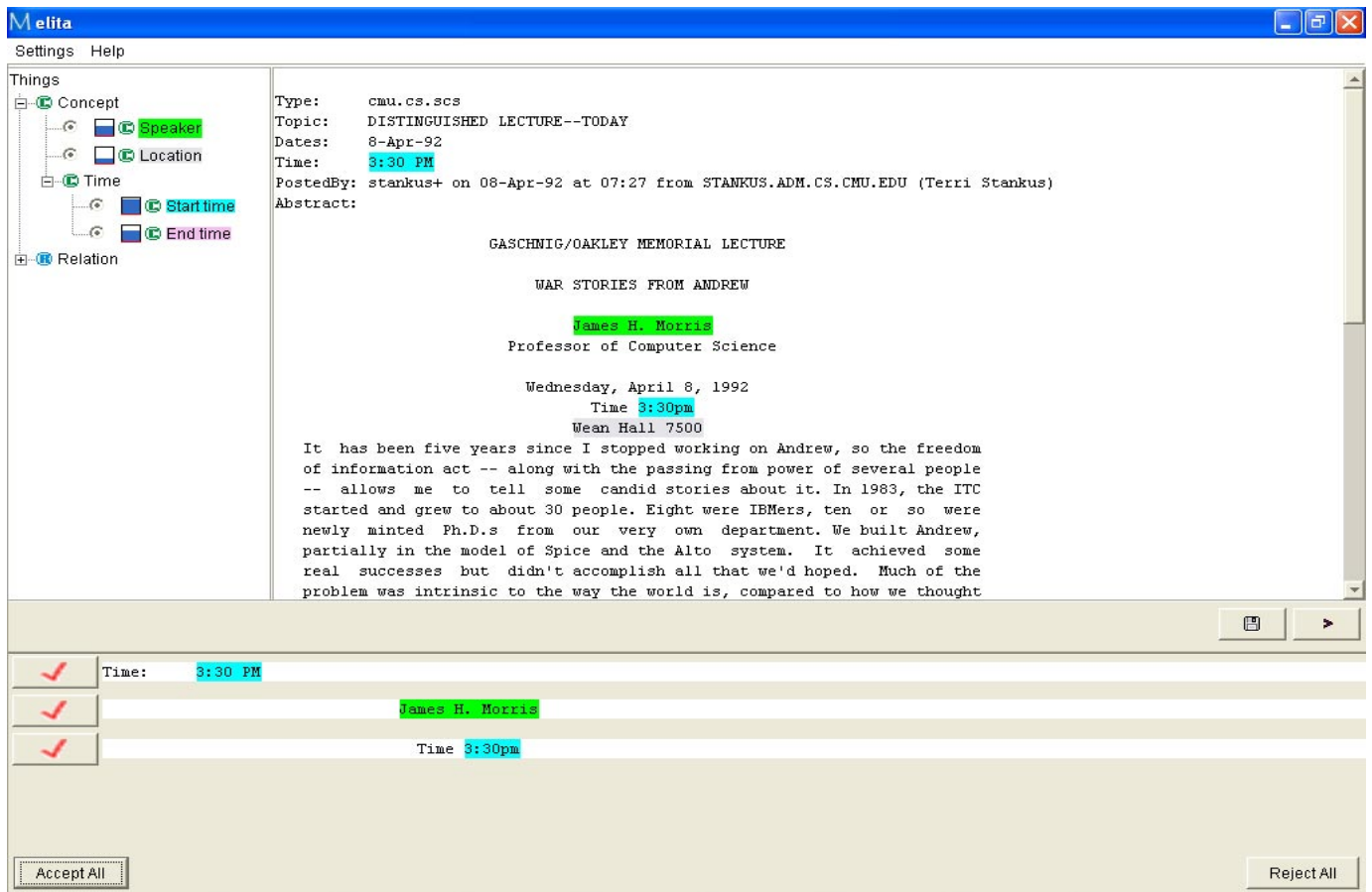
Melita does not differ in appearance from other annotation interfaces such as the Gate annotation tool, or MnM or Ontomat. This is because – as mentioned – it is a demonstrator to show how a typical annotation interface could interact with the IES. The novelty of Melita is the possibility of (1) tuning 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. Suggestions can be presented in the suggestion area or in the document area according to a number of criteria. When presented in the suggestion area an explicit selection (on the tick box) is required to the user to accept the suggestion, otherwise the suggestion is not inserted. When presented directly into the document under annotation suggestions are displayed using the same colour code (e.g. blue background for speaker), but they are made recognizable as suggestions because of a special coloured border. The assumption here is that annotations are considered correct unless the user removes them explicitly. The presentation strategy adopted displays unstable tags (i.e. tags not yet fully reliable) in the suggestion area, while tags considered reliable by the system are displayed directly in the document. Note that reliability is independent for each piece 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 inserted in the document pane while the annotations for the speaker will go into the suggestion panel.

5.1 Controlling Proactivity

Users can customize the behaviour of the IES tuning the level of IES proactivity thus changing the level of intrusiveness by using a special slidebar (fig.4). It allows to set two thresholds that divide
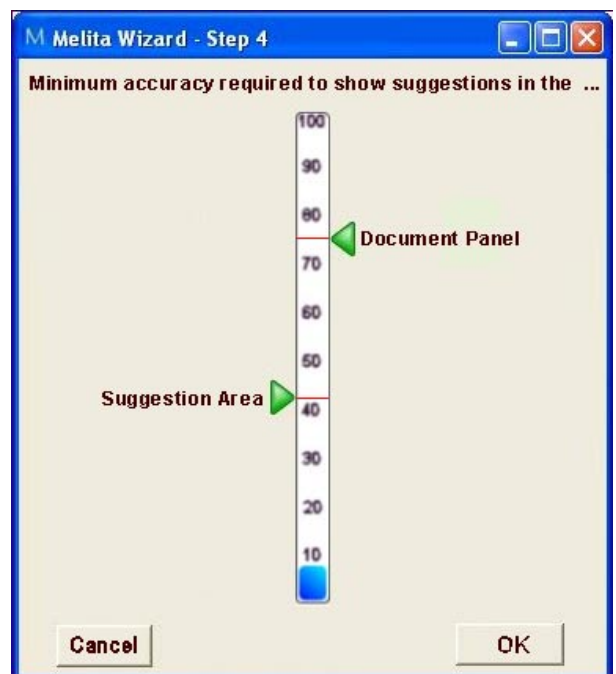

Figure 4: the slidebar for tuning system's intrusivity

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 inserting annotation in the suggestion panel. The second

threshold defines the minimum accuracy the system must reach before starting suggesting in the document panel. In the example in figure 4 the system will suggest in the suggestion panel when its accuracy is between 43 and 75% and in the document panel when greater than 75%. When accuracy is less than 43% the IES does not suggest (i.e. it is still in training mode). This general default holds for all the nodes in the ontology, but it can be overridden for specific tags by using the same kind of window. Changing the default for specific tags is useful because users can have different feelings about intrusiveness for different kinds of information depending on the effort required to identify and select that piece of information. It is worth noting that the same slidebar shows the accuracy currently reached by the IES for the specific information: it is the blue filler mark that grows from the bottom (around 10% in figure 4). It is a feedback on the current status of the IES, e.g. if it is in training mode, if it is suggesting in the suggestion panel, etc. Moreover such feedback should support an intuitive changing of the current IES behaviour, e.g. turn off the IES suggestions by lifting up the two arrows beyond the blue maximum level. Note that the same information is presented near each node in the ontology panel: a small square is divided in three parts (corresponding to the three areas above). The small square fills in the same way the slidebar fills. In this way the user has always a feedback on the current status for each piece of relevant information.

6. 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. Such corpus was already used for evaluating a number of adaptive algorithms [10]. In our experiment the annotation in the corpus was used to simulate human annotation in the methodology described above. 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 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[2], 47 wrong, leading to Precision=84 Recall=61. 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 80% f-measure is reached only after about 100 texts. 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. Note that in the

experiment we did not use a Named Entity Recogniser. A NERC would have allowed reducing the needs of examples for speaker. We performed the same type of analysis on other corpora such as the Austin TX Jobs announcement corpus and found similar results.

6.1. Is it Worth Using IE?

The 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. 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., 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 the complex cases.

Tag	Amount of Texts needed for training	Prec	Rec
stime	30	91	78
etime	20	96	72
location	30	82	61
speaker	100	75	70

Table 1: The amount of training texts needed for reaching at least 75% precision and 50% recall

7. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a modality of interaction between an adaptive IES and a classical annotation interface for the Semantic Web. We have defined a modality in which the interface and the IES cooperate in order to obtain effective annotation in the way preferred by a specific user. We have also explained how to organize learning in order to reduce or avoid any idle time from the user point of view. Then we have discussed how it is possible to maintain a reasonable timeliness in learning from examples while hiding to users the delay necessary for training the underlying IES. Finally we have presented Melita, a demonstrator that implements such methodology and we have described how user configurations in Melita are turned into settings for Amilcare.

We believe that this methodology of interaction between the IES and the annotation interface allows to fully exploiting the potentiality of adaptive IE for annotating texts because:

1. It inserts in the usual user environment without imposing particular requirements on the annotation interface used to train the IES. (2)

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

---

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

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 tunable and therefore easily adaptable to the specific user needs/preferences.

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 stop or slow down the annotation process).

Future work will consider the better formalization of the way in which Melita's settings are turned into IES settings. The currently adopted solution is still under evaluation and it needs further development and experiments, as currently it is completely arbitrary and the risk is to produce an opaque effect on the user with respect to the way in which the IES is influenced.

8. ACKNOWLEDGEMENT

Bibliography

[1] M. E. Califf, D. Freitag, N. Kushmerick and I. Muslea (eds.): AAAI-99 Workshop on Machine Learning for Information Extraction July 19, 1999, Orlando Florida (http://www.isi.edu/~muslea/RISE/ML4IE/)

[2] F. Ciravegna, R. Basili, R. Gaizauskas (eds.) ECAI2000 Workshop on Machine Learning for IE, Berlin, 2000, (www.dcs.shef.ac.uk/~fabio/ecai-workshop.html)

[3] F. Ciravegna, N. Kushmenrick, R. Mooney and I. Muslea (ed.), 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 (http://www.smi.ucd.ie/ATEM2001/)

[4] J.B. Domingue, M. Lanzoni, E. Motta, M. Vargas-Vera and F. Ciravegna: "MnM: Ontology driven semi-automatic or automatic support for semantic markup", submitted paper.

[5] BADGER Information Extraction (IE) Software, http://www-nlp.cs.umass.edu/software/badger.html

[6] S. Handschuh, S. Staab and F. Ciravegna: "S-CREAM - Semi-automatic CREAtion of Metadata", submitted paper.

[7] C. A. Thompson, M. E. Califf, and R. J. Mooney: "Active Learning for Natural Language Parsing and Information Extraction", *Proceedings of the Sixteenth International Machine Learning Conference* (ICML-99), Bled, Slovenia, pp. 406-414, June 1999.

[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 (IJCAI-01), Seattle, August, 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, 2002, forthcoming.

[10] F. Ciravegna: "Adaptive Information Extraction from Text by Rule Induction and Generalisation" in Proceedings of *17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*, Seattle, August 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, IJCAI-97.

[13] R. S. Mickalski, I. Mozetic, J. Hong and H. Lavrack: *The multi purpose incremental learning system AQ15 and its testing application to three medical domains'*, in Proceedings of the 5th National Conference on Artificial Intelligence, Philadelphia: Morgan Kaufmann.

[14] F. Ciravegna: "Challenges in Information Extraction from Text for Knowledge Management", *IEEE Intelligent Systems and Their Applications*, November 2001.

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

[16] A. Douthat, "The message understanding conference scoring software user's manual", in [17]

[17] 7th Message Understanding Conference Proceedings, MUC-7. http://www.itl.nist.gov/iaui/894.02/related_projects/muc/