Incremental Formalization of Document Annotations through Ontology-Based Paraphrasing

Jim Blythe and Yolanda Gil
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
+1 310 448 8251
blythe@isi.edu, gil@isi.edu

ABSTRACT
For the manual semantic markup of documents to become widespread, users must be able to express annotations that conform to ontologies (or schemas) that have shared meaning. However, a typical user is unlikely to be familiar with the details of the terms as defined by the ontology authors. In addition, the idea to be expressed may not fit perfectly within a pre-defined ontology. The ideal tool should help users find a partial formalization that closely follows the ontology where possible but deviates from the formal representation where needed. We describe an implemented approach to help users create semi-structured semantic annotations for a document according to an extensible OWL ontology. In our approach, users enter a short sentence in free text to describe all or part of a document, and the system presents a set of potential paraphrases of the sentence that are generated from valid expressions in the ontology, from which the user chooses the closest match. We use a combination of off-the-shelf parsing tools and breadth-first search of expressions in the ontology to help users create valid annotations starting from free text. The user can also define new terms to augment the ontology, so the potential matches can improve over time.

Categories and Subject Descriptors
H.3.1 [Information Storage and Retrieval] Content analysis and indexing, *abstracting methods* I.2.6 [Artificial Intelligence] Learning, *knowledge acquisition*

General Terms
Human Factors, Design.

Keywords
Document annotation, semantic markup, knowledge acquisition.

1. INTRODUCTION
Semantic annotations of documents can help to qualify their contents, enable search and retrieval, and to support collaboration. In some approaches, these annotations are extracted automatically from the document [Dill et al., 5]. In other approaches, the annotations are manually created by users [Kahan and Koivunen, 11]. Handcrafted annotations may be more accurate but more importantly they enable users to reflect their opinions or their own analysis of the document. However, expressing these annotations formally is difficult for most web users, who are not skilled knowledge engineers and may be unfamiliar with the domain terms used in the formal annotations. Helping users to create correct formal annotations that capture their intended expression is a challenge that must be addressed if semantic annotation tools are to become widely accessible.

In this paper we describe an approach that accepts user annotations as short statements of free text and then helps to formalize the statement, partially or totally, by mapping it to an existing schema or ontology. Given a free text statement, our implemented system, called ACE[1], creates plausible paraphrases of the sentence generated using the ontology and presents them to the user as possible canonical forms of their original statement. If new terms appear in the statement, ACE will suggest to the user possible extensions to the ontology that incorporate the new terms. To generate the plausible paraphrases, the system makes use of a parser and a beam search of expressions within the ontology. Our implementation draws from ontologies in OWL [OWL, 15], but can easily be applied to other mark-up languages, such as RDF schemas.

Our work extends the TRELLIS annotation tool that enables users to express their analysis of possibly contradictory information sources [Gil and Ratnakar, 8]. In TRELLIS, each statement in the analysis is formulated in free text, and linked to other statements through a set of domain-independent formal constructs for argumentation, expressed in a semantic markup language. TRELLIS is an interactive tool that helps users annotate the rationale for their decisions, hypotheses, and opinions as they analyze information from various sources. ACE, described in this paper, extends TRELLIS by helping users to formalize the text statements incrementally according to a domain ontology that can be extended during this interaction.

The main contributions of this work are:

- an implemented annotation tool that allows the user to create formal annotations incrementally by interacting with paraphrases and without having to read the annotation in its formal language.
- the use of search techniques in an ontology to provide plausible fragments of a formal annotation that match the user's terms.

In the next section we provide a short overview of the TRELLIS system, followed by a detailed example of using ACE to make statements more formal and more amenable to matching. We then present a preliminary evaluation to explore the contributions of

WWW 2004, May 17–22, 2004, New York, New York, USA.
ACM 1-58113-844-X/04/0005.

[1] Annotation Canonicalization through Expression synthesis.

the different technologies in ACE to the task of matching related statements. We end with a discussion of the system and a comparison to related work.

2. OVERVIEW OF TRELLIS

In a wide range of decision-making tasks, such as choosing a vacation destination, family history research or design decisions, the decision maker must keep track of a number of information sources and maintain a succinct description of each one as well as a record of how they are combined to support various conclusions. The goal of TRELLIS is to enable users to create annotations of their analysis of alternative sources of information as they make a decision or reach a conclusion based on their analysis. Once this rationale is recorded, it can be used to help users justify, update, and share the results of their analysis. Users need support after they have made a decision, reached a conclusion, or made a recommendation, since they are often required to: 1) explain and justify their views to others, 2) update the decision in light of additional information or new data, 3) expose the intermediate products of the final recommendation to others that may be analyzing related information to make similar decisions.

A *statement* in TRELLIS is a piece of free text information or data relevant to an analysis. A statement may have been extracted, summarized, or concluded from a document.

For example, the following news item might be summarized with the statement "West Ham may sign Marcus Bent":

> *West Ham boss Glenn Roeder is bidding to end the club's Premiership woes with the signing of Ipswich striker Marcus Bent in the new year, according to a report. Ipswich had originally stamped a £3m price tag on the front man. But now the club looks set to drop to £2m to bring in money for manager Joe Royle's January signings, according to the Sun newspaper.*

Other summaries are also possible, depending on what the user views as salient information in the text.

TRELLIS helps the user to construct an argument for or against some conclusion by combining statements like these with a set of standard constructors such as 'is supported by', and 'in contrast with'. Figure 1 shows an application of TRELLIS to analyze a potential transfer in a sports domain. On the left hand side, the user provides the conclusion of the argument and lists sources, which may have live URLs, and statements based on the sources. On the right side, the user has constructed an argument in support of the conclusion from the statements. More details on TRELLIS can be found in [Gil and Ratnakar, 9].

Since the statements in TRELLIS are represented in free text, there is limited opportunity to provide inferential support or to locate related statements and analyses from other users who may use different phrasing. ACE, described in this paper, extends TRELLIS by helping users to formalize the text statements incrementally according to a domain ontology expressed in OWL.

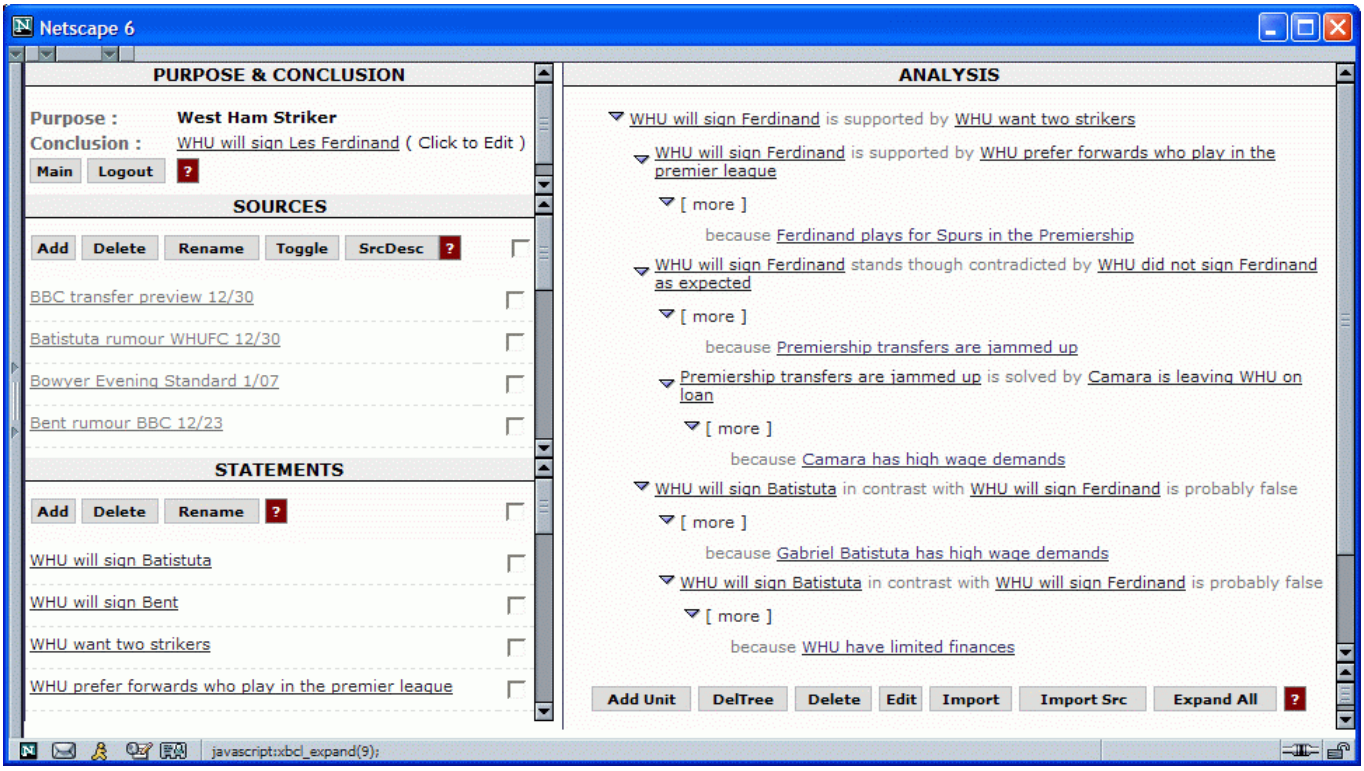

Figure 1: TRELLIS is used to evaluate the likelihood of a possible event in a sports transfer domain.

.

2.1 Scenario

We illustrate ACE with the scenario used above, drawn from professional sports. Teams often sign players amidst controversy and rumors, reflected by press articles with dissenting views as well as many on-line discussions of opinionated fans. Here, a user may want to annotate a certain news item, for example with his conclusion reached after reading it that a certain team is very likely to sign a certain player. Consider a conclusion, for example, that a particular football club, West Ham, wishes to sign attacking players who are currently playing in the top league in that country, the English Premier League (EPL).

Two users may express this same conclusion using two quite different statements, for example "West Ham are targeting strikers from the EPL" and "WHU prefer forwards who play in the Premier League". It is not our aim to match pairs of phrases like these in all cases — such a task would require a deep understanding of the sentences that is beyond the state of the art. However, even partial reformulations of the sentences would be useful if they help expose their similar meanings. This will improve the likelihood that a search engine can detect the similarities of both analyses. Thus, ACE's task is to suggest reformulations of a concise text statement that conform as closely as possible to the desired ontology or schema.

ACE brings together four techniques to help with this task:

- First, Term Replacement performs a substring match on the sentence against the terms defined in the ontology and suggests re-writing specific terms with their canonical values. For example, in the second sentence above, the tool might suggest to replace "forwards" with "strikers" based on the known synonyms of that class.
- Next, the Parser generates information about the sentence that can help simplify it, for example to find determiners or passive verbs.
- Next, the Expression Composer makes use of the ontology again to search for plausible compositions of relations and classes that can link the matched terms.
- Finally the user can create new terms in the ontology at any time in the process. To do this, the user highlights a word in the current annotation and chooses to define it as a new term. A tree widget allows the term to be placed within the hierarchy, and it is then available to all the other modules. For example, the user may choose to add a new team or a new kind of player with this tool.

We describe the first three techniques in order below. Figure 2 shows the architecture of ACE.

Our design of ACE is guided by two principles. First, we aim for an interactive system that leaves the user in complete control of the process. At each step, ACE makes suggestions to the user rather than reformulating the sentence automatically. This process may be partial, leaving part of the sentence unconverted and generating an annotation that includes some text as well as some expressions generated in the markup language. Our second design principle is to use the component modules, such as the parser and expression composer, in ways that are robust to the potential failures of the modules on free text. For example, the use of the parser is robust in the sense that reformulations can be suggested even if the tool fails to parse the sentence or returns an incorrect parse.

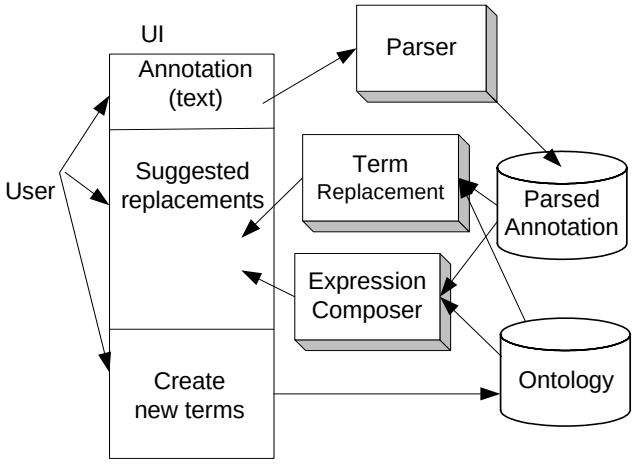

Figure 2: ACE architecture

In this section we discuss the scenario described earlier: two statements are entered in a sports domain: "*West Ham are targeting strikers from the EPL*" and "*WHU prefer forwards who play in the Premier League*." We show how they can be mapped to the same partially formalized expression, in which parts of the original text have been re-expressed within the ontology and parts remain as free text. Figure 3 shows the suggestions that are made after the term-replacement step, and Figure 4 shows the suggestions that result from the latter steps. In the first step, synonyms for simple terms in the ontology are replaced using a sub-string match. This step contributes to putting the sentence in a regular form, but another purpose is to confirm some of the entities in the domain with the user. Next, the tool attempts to parse the sentence if possible, to remove words that are not processed when finding candidate formal statements. Finally we search for plausible compositions of relations and terms in the ontology that match terms and other words found in the user's sentence. Below we describe these steps in detail.

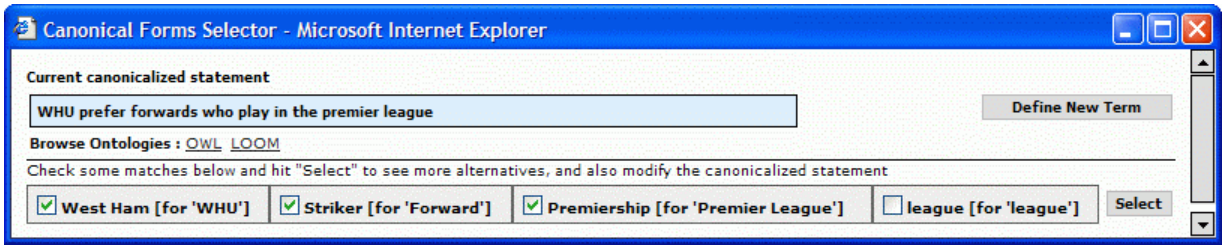

Figure 3: ACE suggests primitive term replacements in the text based on the underlying ontologies.

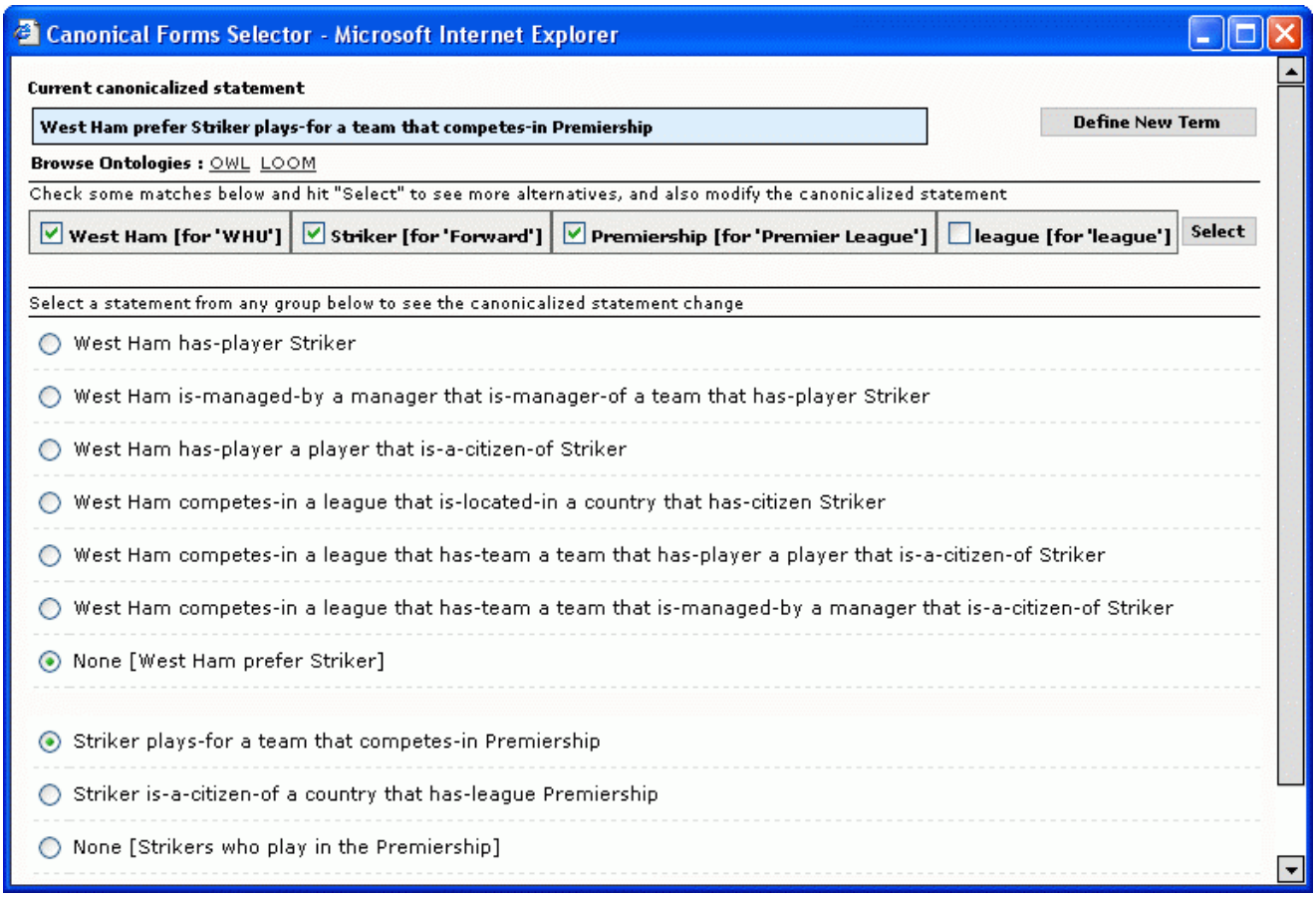

Figure 4: After primitive terms have been verified, ACE suggests reformulations of the original text through the Expression Composer. The phrases are automatically generated from paths in the ontology. Choosing a path can help to disambiguate the text, for example 'Striker from the Premiership' could be replaced either with 'Striker plays-for a team that competes-in Premiership' or 'Striker is-a-citizen-of a country that has-league Premiership'.

2.2 Term Replacement

In the first step, synonyms for simple terms in the ontology are replaced using a substring match. While this step contributes to putting the sentence in a regular form, another purpose is to confirm with the user some of the known entities from the domain ontology found in the sentence. The suggested term replacements shown in Figure 3 are generated from the ontology in Figure 5, using hand-coded synonyms augmented by synonyms suggested by WordNet [Fellbaum, 6]. In our ongoing example, when the user confirms the three substitutions, the tool can continue in the knowledge that the sentence contains a particular team (West Ham), a generic striker which is a subclass of player, and an instance of a league (the Premiership). Relations and event templates in the ontology are typed, and this information about domain entities will be used to search for compositions of events and relations that match those types. The information is also used to aid the next step, parsing the sentence, by replacing compound terms that the parser may not recognize with generic pronouns that are easier to parse.

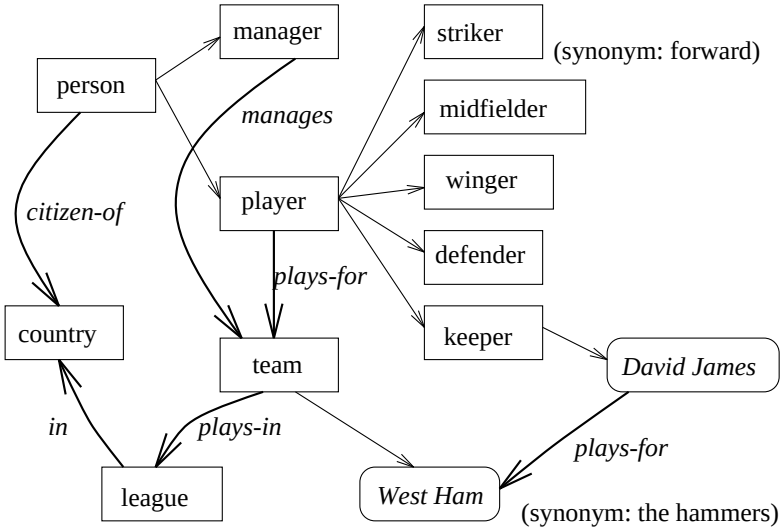

Figure 5: Fragment of the soccer ontology showing some of the suggested terms and their relationships. Each relation has a named inverse, so that paths built by the expression composer can traverse links in either direction

2.3 Parsing to Improve Sentence-Level Matching

ACE makes use of a parse of the sentence, if one can be made, to improve the power of term matching and expression search by making the sentence structure more simple. For example, words identified as determiners are removed during matching and re-inserted in the suggested reformulations, for instance in the sentence "*They want two strikers*", the word "*two*" will be ignored during matching. The same approach can be taken with negation. For example, in "*Liverpool did not sign Ronaldo*", the parser allows us to perform matching on the sentence without the negation and re-insert it in the re-formulated version suggested to the user.

As another example, conjunctive sentences can be split up and treated separately. For instance the sentence, "*They want two strikers but have limited funds*" can be treated as "*They want two strikers"* and "*They have limited funds"*. The parse also identifies subject, object and verb information in the sentence and noun plurality. This information is used in matching of event templates, described below. We currently use a probabilistic parser available from the JavaNLP project [Klein and Manning, 12] that provides a tree-structured parse. For example, the parse of the sentence above is shown in Figure 6.

2.4 Expression Composer: Searching to Suggest Compound Expressions

Finally we search for plausible compositions of relations and terms in the ontology that match terms and other words found in the user's sentence. A forward beam search is made through the space of valid compositions of expressions, made up of relations, classes, instances and event templates. The search returns the shortest expressions that include a set of requested words, possibly including synonyms for the terms. It then generates a sentence encoding the expression for the user to consider. If no expressions match all the requested words, paths are used that match are subset of the words, weighted according to how many words are matched and whether synonyms are used. This approach was originally applied to help users build complex expressions of problem-solving knowledge, as described in [Blythe, 2].

For example, in the sentences above, matched terms include "striker", a kind of player, and "the Premiership", an instance of a league. Since the ontology includes the facts that players play for teams and that teams are organized in leagues, one suggestion the tool makes is to replace *strikers from the EPL* with *strikers who play for a team that plays in the EPL*. If there are several such paths linking the terms in the ontology, a number of the shortest paths will be suggested. Figure 4 shows the options that are generated from a small ontology for the example sentence "W*HU prefer forwards who play in the Premier League*" after term replacement.

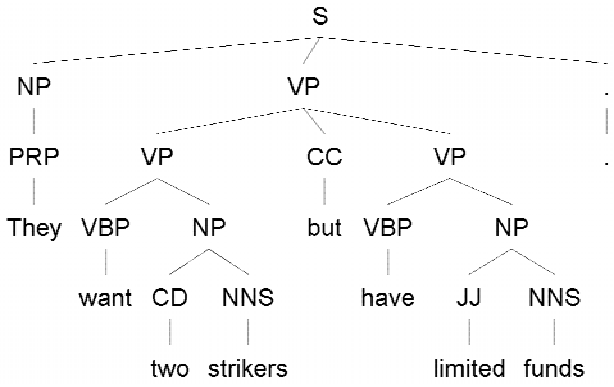

Figure 6: An off-the-shelf parser can help identify compound sentences and cardinality.

Notice that the system is disambiguating the text. For example, the phrase "*players from the Premier League*" might refer to players who play in the premier league now, or who have been transferred from there, or who were born in the same country. If these relations are captured in the ontology, they will be presented to the user as alternatives to choose from.

Users can also add terms to the ontology by selecting a portion of the statement and choosing where the new term should be inserted in the class hierarchy. Currently only classes are added; instances and relations will be included in future versions.

The expression composer represents a different and complementary approach to the term replacement module. The latter takes its lead from the user's sentence, and will not suggest to introduce a term unless there is at least a substring match with the sentence. The expression composer, on the other hand, performs search in the domain ontology based on the introduced terms, using the input sentence only to weight the results. Although this can sometimes produce results that are surprising to the user, one advantage is that terms in the ontology that might be closely related to the concepts the user is expressing are likely to be presented even though they do not have a surface match with the user's chosen words. Similarly, since the composer does not directly use the words in the sentence, it can make suggestions even if few or none of the words in the sentence are recognized apart from the matched terms.

3. DISCUSSION

We have described ACE, a tool to help users create document annotations that may include complex expressions based on an underlying ontology. ACE uses off-the-shelf parsing and a beam search in order to suggest compound expressions from an underlying ontology that may match the user's free text input. The system is designed to be robust, allowing partial formalizations of the annotation and not relying on a successful parse of the user's input.

3.1 Related Work

Annotea [Kahan and Koivunen, 11] provides an open framework for RDF-based web annotations. Annotea is neutral about the user interface and the work we describe is complementary, showing how users can create complex formal annotations with an interactive user interface. An interface integrated with Amaya [Amaya, 1] is provided that focuses on attribute-based annotations. In contrast, ACE helps users create quite complex expressions involving compositions of relations.

Melita [Ciravegna et al., 4] is an interactive annotation tool that makes use of a separate training phase to learn annotation rules that are used to make suggestions to user for subsequent texts. The rules map from the document text itself into terms within the ontology that fill pre-defined overall document patterns (for example a talk at some time and location). Like Melita, S-CREAM is based on the information extraction component Amilcare [Handschuh et al., 10] as is MnM [Vargas-Vera et al., 20]. Both learn knowledge-extraction rules to suggest annotations to the user. However, none of these tools suggest compound expressions based on the ontology, as our system does using search, and therefore they can only map the text to pre-defined structures.

Other researchers have used parsers to process short sentences in order to enter information in a knowledge base. For example, Chklovski [Chklovski, 3] uses the Link Grammar parser [Sleator and Temperly, 18] to create a sentence 'signature' which, among other things, removes determiners and closed-class words. All subsequent processing is then performed on signatures. In contrast, we use both the parse and the original sentence. The parse is used to modify the way the expression composer is used, while features of the original sentence, such as determiners, are restored when suggestions are made. It would be possible in principle to use signatures with our approach, and we plan to investigate this.

3.2 Discussion and Future work

The use of search is central to our approach and this affects how well ACE scales. Our experiences in a military planning domain [Blythe, 2] indicate that the tool scales well as the ontology size increases. Here, the ontology contains around 100 concepts and several hundred relations but the search is typically completed within a few seconds, adequate for interactive work. However, the current search approach does not scale well with the length of the smallest matching expression, since in the worst case it considers an exponentially growing number of candidates. In practice, we use a small beam of about 20 candidate examples and run the search with a time limit of 10 seconds of real elapsed time in this domain, returning without a match if none can be found within the time limit.

We are currently investigating a dynamic programming technique that improves on this performance. This technique uses a hyper-graph whose nodes correspond to the data types in the domain and each of whose links corresponds to a set of relations, linking a set of input nodes to an output node the input nodes match the relations' domain, and the output node is included in their range. On each iteration, the algorithm stores new candidate compound terms at each node by following the links and using the terms developed in the previous iteration. The approach is complete and will guarantee to find the smallest solution. It is more efficient than the beam search we have used previously because it facilitates aggregating terms to reduce the search space, as well as other pruning techniques. We have implemented this approach within ACE and are exploring the time-space tradeoffs.

In our experiences, ACE frequently finds a desired match as one of a small number of suggestions. We plan user evaluations in several domains to test the generality of the results. The names used for relations in an ontology affect both the matches returned and how understandable the users find their descriptions, so we will evaluate both with ontologies designed by ourselves and by others. The tool can be used with any OWL ontology, and we have encouraging initial results with the planning domain, that uses terms defined by several different groups.

One area for future work is to gain a better understanding of the behavior of the system with a small ontology, or with a shallow ontology with fewer than usual relations but many instances, or with an ontology built for a related domain. As communities of users begin to use TRELLIS, we also intend to explore analyses that bridge two or more ontologies and form a link between the respective communities. We are also using a variant of the approach to provide guidance to a task management system that supports an office assistant tool, with a growing ontology designed by several groups in collaboration.

Another direction is to make greater use of NLP tools during reformulation of annotations, including WordNet [Fellbaum, 6], stemmers [Porter, 16] and parsers [Klein and Manning, 12]. While ACE currently uses synonyms provided by WordNet to improve its matches during search, further improvements may be possible

using a word's gloss or a semantically enhanced version such as Extended WordNet [Mihalcea and Moldovan, 14]. The existing parse of the original annotation can also be used more fully, for example matching expressions can be modified using the parse to conform more closely to the user's original expression.

3.3 Contributions

This paper makes the following contributions:

1. We take a closed-loop approach to the interactive extension of ontologies in the context of a task that uses them, in this case an annotation task. Previous work on ontology editors [Gennari et al., 7, McGuinness et al., 13] assumes that the user's sole task is to extend the ontology, leaving their purpose and use outside the reach of the editors. Such an open-loop approach makes it hard to ensure that the final ontology will be suitable for its purpose.
2. We take a more flexible approach to controlled grammar interfaces, where the user provides free text input that is matched to the grammar. This simultaneously lowers the barrier for users to enter statements in the grammar and allows a partial match, where some input is matched and some remains as free text. There is an incremental payoff for the user's effort invested in formalizing statements. Previous approaches are either completely formal [Gennari et al., 7, McGuinness et al., 13] or they are completely informal, using unprocessed text [Stork, 19, Singh and Barry, 17].

We also take a more dynamic approach to controlled grammar interfaces, where the grammar is generated from ontologies that the user continues to extend. Previous work on controlled grammars assumes that the user's input must comply with a pre-engineered grammar that is to be adopted as a standard, e.g. [Wojcik, 21].

4. ACKNOWLEDGMENTS

We are grateful to Tim Chklovski for useful discussions of this work and his comments on draft versions. Varun Ratnakar implemented the interface and integrated the tool with Trellis.

5. REFERENCES

[1] Amaya 03. http://www.w3.org/Amaya.

[2] Blythe, J. Integrating expectations from different sources to help end users acquire procedural knowledge. *Proceedings of IJCAI'01* (Seattle, WA, August 2001).

[3] Chklovski, Y., Using Analogy to Acquire Commonsense Knowledge from Human Contributors, Ph.D. Thesis, MIT Artificial Intelligence Laboratory technical report AITR-2003-002, 2003.

[4] Ciravegna, F., Dingli, A., Petrelli, D. and Wilks, Y., User-System Cooperation in Document Annotation based on Information Extraction. *Proceedings of EKAW'02*. (2002).

[5] Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., Jhingran, A., Kanungo, T., Rajagopalan, S., Tomkins, A., Tomlin, J., and Zien, J. SemTag and Seeker: Bootstrapping the Semantic Web via Automated Semantic Annotation, *Proceedings of WWW12*, Budapest, 2003.

[6] Fellbaum, C., Ed. *WordNet, an Electronic Lexical Database*, MIT Press, 1998.

[7] Gennari, J., Musen, M., Fergerson, R., Grosso, W., Crubezy, M., Eriksson, H., Noy, N., Tu, S. The Evolution of Protege: An Environment for Knowledge-Based Systems Development *International Journal of Human-Computer Studies*, 58(1), 2002.

[8] Gil, Y. and Ratnakar, V. Trusting Information Sources One Citizen at a Time. *Proceedings of ISWC'02*. (2002a).

[9] Gil, Y. and Ratnakar, V. TRELLIS: An interactive tool for capturing information analysis and decision making. *Proceedings of EKAW'02*. (2002b).

[10] Handschuh, S., Staab, S. and Ciravegna, F., S-CREAM: Semi-automatic CREAtion of Metadata. *Proceedings of EKAW'02*. (2002).

[11] Kahan, J. and Koivunen, M., Annotea: An Open RDF Infrastructure for Shared Web Annotations, *Proceedings of WWW10*, Hong Kong, 2001.

[12] Klein, D. and Manning, C. Fast Exact Inference with a Factored Model for Natural Language Parsing. Advances in Neural Information Processing Systems 15 (NIPS 2002), 2002.

[13] McGuinness, D., Fikes, R., Rice, J., Wilder, S. The Chimaera Ontology Environment, *Proceedings of AAAI 2000.*

[14] Mihalcea, R. and Moldovan, D. eXtended WordNet: Progress Report, in *Proceedings of NAACL Workshop on WordNet and Other Lexical Resources*, Pittsburgh, PA, 2001.

[15] OWL 03. http: //www.w3.org/TR/owl-features/.

[16] Porter, M., An algorithm for suffix stripping, *Program*, **14**(3) :130-137.

[17] Singh, P. and Barry, B. Collecting Commonsense Experiences, *Proceedings of KCAP'03*, 2003.

[18] Sleator, D. and Termperley D., Parsing English with a link grammar, *Proc. International Workshop on Parsing Technologies*, 1993.

[19] Stork, D. The Open Mind Initiative, *IEEE Expert Systems and Their Applications*, May/June 1999.

[20] Vargas-Vera, M., Motta, E., Domingue, J, Lanzoni, M., Stutt, A. and Ciravegna, F. MnM: Ontology Driven Semi-automatic and Automatic Support for Semantic Markup, *Proceedings of EKAW'02*. (2002).

[21] Wojcik, R. The Boeing Simplified English Checker, 2002, http://www.boeing.com/assocproducts/sechecker