ADVANCES IN DIFFICULT ASPECTS OF REFERENCE RESOLUTION: WORKING NOTES

Marjorie McShane

Working Paper 01-09

November 18, 2009

Institute for Language and Information Technologies
University of Maryland Baltimore County

1 The Nature of this Report

This document is a working paper that is a condensed and slightly modified version of the project report (McShane 2009b) submitted to NSF at the close of the 2008-2009 SGER entitled "Advances in Difficult Aspects of Reference Resolution." It excludes programmatic aspects of the project report and excludes portions of data and analysis that are not needed for an understanding of the overall work.

2 Introduction

The automatic resolution of the pronouns *this, that, it* is more difficult than the resolution of the referring expressions more typically treated by NLP systems, such as *he* and *she*. *This/that/it* provide practically no semantic features about their sponsor, they can refer to entities of any ontological type (object, event, property or any combination of these), and their sponsor need not be an NP, it could also be a span of text of any size. There are no annotated corpora of any reasonable size that cover these referring expressions as used in all of their semantic functions.

Our **hypothesis** is that for difficult referring expressions, whose treatment has eluded traditional stochastic systems, a linguistically informed approach is warranted. The **approach**, in a nutshell, is as follows. People create an inventory of features and value sets that they believe will be useful in automatic reference resolution. They then create and manually score combinations of feature values, with the scoring reflecting how frequently a candidate sponsor having those feature values is expected to be the actual sponsor. We currently use a four-level scoring process, though one could choose any grain size of distinction. Finally, they create an inventory of scored feature value combinations to be used in a reference resolution engine that matches the features of each candidate sponsor with the inventory of scored feature value combinations. The candidate whose feature values match the highest scoring feature value combination is selected as the sponsor.

The description thus far makes no reference to anything computational; indeed, one could create a reference resolution engine according to the approach suggested here without any empirical evidence at all. To put a finer point on it, the **theoretical**, **pre-computational** (pre-applied) **nature of the work** -- which centrally involves combining evidence in linguistically supported ways and assigning confidence levels to the combined evidence prior to text processing – requires no empirical or statistical support. The theory, however, can be **implemented** in many ways, using various kinds and degrees of empirical support. For example, annotating a corpus and using it as a search space would provide developer-linguists evidence for combining feature values and scoring those combinations (this is, in fact, what we have done): the larger the corpus, the better one might expect the results of this process to be. Or, if an annotated corpus were large enough, it could serve as the substrate for statistical processing that would suggest predictive combinations of feature values and their scores. Alternatively, one of the many middle grounds between these two approaches would be for people to posit the combinations of feature values and a statistical engine to determine their scoring. The main point is, although we would expect a larger annotated corpus to provide better evidence for the creation and scoring of feature value combinations than a smaller corpus

or no corpus at all, the corpus can actually be of any size or be non-existent, which removes a major roadblock to the treatment of difficult referring expressions that has been imposed by the statistical paradigm.

3 Carving Out A Problem Space

The resolution of *it/this/that* has not been pursued much to date presumably because some instances of these referring expressions are, indeed, very difficult to resolve automatically. However, that does not mean that *all* instances are difficult to resolve, and treating all cases as if they were the worst case is not useful or necessary.

Text processing is typically carried out in pipeline fashion: tokenization, POS labeling, recognition of proper names, syntax, semantic disambiguation and the establishment of basic semantic dependencies, and finally the analysis of reference relations, indirect speech acts, relative expressions of time and place, and the like. Of course, not all systems undertake all of these stages of processing. A key to treating *it/that/this* is understanding that not all reference resolution needs to be postponed until all of those other steps have been completed. For example, when *it* occurs in an existential construction like "It is a dog", 'dog' constrains the interpretation of 'it', so this 'it' is not just a garden-variety, uninformative 'it', it is a lightly cloaked 'dog'. Similarly, when any of these REs is used in an idiom or phrase – e.g., *Far be it from me to criticize you* – then the 'it' does not need to be resolved at all. To repeat, not all instances of *it/this/that* need to be treated by a monolithic, late-stage reference resolution engine – that is, as long as reference is being resolved in an end-to-end system and not as an isolated task (see McShane et al. 2009b).

The OntoSem environment in which our group works (which is the implementation of the theory of Ontological Semantics; Nirenburg and Raskin 2004), carries out all of the abovementioned aspects of text processing without a strict pipeline. We are developing the theory of reference resolution reported here using the OntoSem environment as a conceptual substrate, though the approaches being developed are not constrained to OntoSem. For more information on OntoSem, see, e.g., McShane et al. 2004a,b, 2005a,b, 2008a,b, 2009a.

We must emphasize that while our goal is *not* to treat only the simple cases using only light heuristics, we also do not believe that full and accurate semantic and pragmatic interpretation of every text is required as a prerequisite for treating these REs, and we do not believe that it is wise to leave all reference resolution until a single "reference resolution" stage that is handled by an all-purpose algorithm. As such, we have been developing concrete means of detecting and resolving the more straightforward cases before resorting to the big guns for the most difficult cases.

4 Simplifying Assumptions

Certain simplifying assumptions were needed in order to make the problem space feasible for the work reported here.

1. Pleonastic, idiomatic, etc., *this/that/it* were omitted from the corpus, with the understanding that they would have to be automatically detected. A number of systems have been developed to do this automatically.
2. All sponsors had to be in the text, not syntactically elided or available only in the

extra-linguistic context. For English, which does not have argument ellipsis, this is a perfectly reasonable starting point. By contrast, if one were working with a language like Russian or Chinese, one would have to deal with the ellipsis problem from the outset.

3. We excluded from the corpus contexts that didn't make sense, that required extensive preceding context to be interpreted (our reference window was 3 sentences), etc.
4. We dealt with normative text genres, not emails or blogs.
5. The window of reference was the 3 preceding sentences, if there were 3; otherwise, whatever there was.
6. We assumed that text span antecedents had to be adjacent to the sentence containing the RE without skipping over material or containing any gaps. This is an oversimplification, especially in dialog, where various sorts of interjections, side comments, etc., are common. We recorded cases in which the actual text span sponsor was not among our candidates and will study those further in the future.
7. All annotation was carried out by a single person, with no cross-checking or interannotator agreement considerations (though post-doc checking of student work was common, especially during the initial stages of work). It was decided that, for our current goals, having a larger corpus that might contain more errors was preferable to having a smaller corpus with fewer errors.
8. The students who were selecting the sponsors typically did not detect cases in which there was benign ambiguity – i.e., a text span sponsor could be understood to include different spans of text. This aspect of the work will be better carried out by more trained individuals working at a slower pace.
9. For reasons of time, we carried out an abbreviated version of concept mapping (details below).
10. For reasons of time, we carried out our "targeted parsing" annotation only on a portion of the preceding context (see below for details).
11. For reasons of time, we selected only 3 NP non-sponsors rather than all possible ones since each non-sponsor NP needed to be provided with the full inventory of feature values.
12. For reasons of time, students left out contexts that were extremely long or confusing.
13. After a several week long period of training and practice, students were required to annotate a certain number of contexts per hour (on average). Their initial tendency to triple check their work had to be stopped but more errors naturally crept in.

5 Feature Selection and Corpus Annotation: Overview

Feature selection for work on *it/this/that* derived from a combination of the field's past experience with reference-oriented features, the knowledge our environment (OntoSem) could provide from automatic processing, our past work on reference-related issues, and manual analysis of a relatively a small corpus – a few hundred examples of annotated text but a significantly larger amount of text that was not annotated or was only partially

annotated. **The inventory of features does not represent all of the knowledge-based features that can be automatically derived and brought to bear on reference resolution; it simply represents the inventory that we have had time to configure and pursue thus far.**

It must be mentioned in passing that positing features to be used by an NLP system does not imply that these features *explain* linguistic phenomena to the deepest degree. As discussed in McShane 2009c's predictive approach to subject ellipsis in Russian and Polish, the fact that a syntactic (phonetic, semantic, etc.) heuristic is sufficient to predict some elliptical phenomenon does not necessarily make that phenomenon, at base, syntactic (phonetic, semantic); instead, the phenomenon might be ultimately grounded in discourse considerations that happen to have predictable syntactic (phonetic, semantic) reflexes. Practicality dictates that we not resort to discourse features like "the topic of the sentence" unless we have concrete heuristics that permit us to automatically detect what the topic is.

There are three reasons for corpus annotation in this work, even considering that it cannot be extensive enough to support truly representative fully stochastic processing. First, the corpus is the search space for feature combinations that can help to predict the sponsor. Second, analysis of the annotated corpus – patterns and exceptions to them – can suggest new disambiguating features. Third, the annotations act as a proxy for perfect machine processing. It must be emphasized that, as developers of practical systems, we do not expect perfect results of upstream processing and incorporate this into our use of heuristic values. However, optimizing resolution algorithms on perfect-world input still seems the best way to at least start out with the most clear picture of the data and problem space. Our broader methodology is then to incorporate automatic blame assignment into fully automatic processing, which can be at various levels: e.g., simply "syntax was wrong so we got it wrong" or "why syntax was wrong". The latter is more of an evaluation of our system; the former is an evaluation of what needs to be correct, how to best use information, etc.

Types of Feature Values. All features with multiple-choice values have the values 'yes', 'no', 'ambiguous' or 'I don't know'. E.g., "candidate gender is neuter: yes". We did not use the latter two values much in this annotation effort, though they were used in our early annotation effort.

"Levels" of Features. Some features are basic and others are generalizing. For example, although we elicited the specific gender, number, case role, etc., of the RE and each candidate, we also have the generalizing features "genders match", "genders do not match", and so on, since these can be useful in developing more generalized feature value combinations. The generalizing features are derived automatically from the basic features.

Sources of Feature Values in Annotation. Some feature values are provided directly by annotators and others are automatically computed from the directly provided features. A description of how values are automatically calculated is found below.

Sources of Feature Values in Automatic Processing. Features used in this work will need to be provided by processors that include a preprocessor, morpohological analyzer, syntactic analyzer and semantic analyzer. The "discourse" features incorporated are relatively surfacy and are provided by automatically interpreting various types of preprocessing information. The use of different subsets of features and the use of features

with different confidence levels are discussed later in the document. To repeat an important point from above: we only use features that can be automatically detected by OntoSem, with OntoSem being an example of a working system that grounds the selection of features sought.

6 Corpus Compilation

Texts of different genres were included in the corpus: a play by Oscar Wilde; newspaper and other (e.g., Wikipedia) articles primarily from the Web but also from the Wall Street Journal; and interviews, both from the Web and recorded as part of our current dialog-oriented application, Maryland Virtual Patient (e.g., McShane et al. 2008a,b, 2009a). Some of the web materials were selected based on some keywords of interest for the abovementioned application and planned extensions to it – certain diseases, battlefield medicine, participatory medicine, etc. Our environment will cover the knowledge aspects of these domains better than domains at large, making our future testing of our end-to-end automatic reference resolution system more representative as there will be fewer upstream errors due to lacunae in lexical or ontological coverage.

The compiled corpus was run through the Stanford parser – the parser used by our analyzer – in order to find instances of *that/this/it* that were used as pronouns. This excluded, e.g., *this/that* as a determiner in a larger NP. For each potentially interesting case, we extracted the given sentence as well as 3 sentences of preceding context, if 3 sentences were available in the document. A three-sentence window of reference is typical for reference resolution systems and a necessary cutoff point for us, for practical reasons, since all sentences in the window had to be annotated manually. The contexts were saved as HTML files, to be worked on further by people, and as XML files. The HTML files permitted copy-pasting into the annotation interface. Many false positive contexts were returned due to parsing errors such as interpreting the complentizer *that* as the pronoun *that*, so some time was spent weeding out the actual contexts of interest.

7 Corpus Annotation Introduction: Content and Methodology

The annotation process is divided into three parts: one carried out by lightly trained undergraduates, one carried out by students with more linguistic background and/or having undergone more training, and trained linguists, what we call "superusers". Due to resource limitations, each annotation was carried out by one annotator, the judgment being that having more data at this early stage of the work would be more useful than double checking the value of every data point.

In resolving *it/this/that* we must assume from the outset that the sponsor could be, syntactically speaking, an NP or a text span. Although our automatic resolution algorithms operate in large part at the level of semantics not syntax, for practical purposes (it is quite slow to read our automatically generated semantic representations) manual annotation was carried out on text strings.

To support not only corpus annotation but also the configuration and evaluation of reference resolution engines, we developed the REDEE environment, which provides the following features, discussed in more detail below.

- It permits the positing of features and value sets relevant for reference resolution. New features and value sets can be added at any time by end users, with no need for intervention by developers. This means that the annotation interface is automatically updated to reflect the new features and the engines that process the annotation data automatically incorporate the new features.
- It facilitates corpus annotation using a variety of expressive means, including the selection and coloring of text spans, the answering of multiple-choice questions, and the filling in of blanks in forms.
- It automatically generates an inventory of derived features from the primary features provided by annotators. These derived features are as important for the compilation of the reference resolver as are the primary features.
- It supports the creation and management of corpora.
- It permits the positing of feature value combinations, the testing of their predictive power over a corpus, and the exploration of contexts that in some way defy expectations.
- If permits the manual scoring of feature value combinations based on a combination of empirical evidence and linguistic intuitions.
- It permits the compilation of scored feature value combinations to be used as input for a reference resolution system.
- It automatically configures the reference resolution system.
- It automatically evaluates the reference resolver on the basis of a variety of parameters.
- It permits fast, convenient study of reference patterns using a host of interface functions.

As an organizing principle for presenting the actual content of the work, we will follow the partitioning of tasks in the annotation interface, whose main pane is shown in Figure 1.

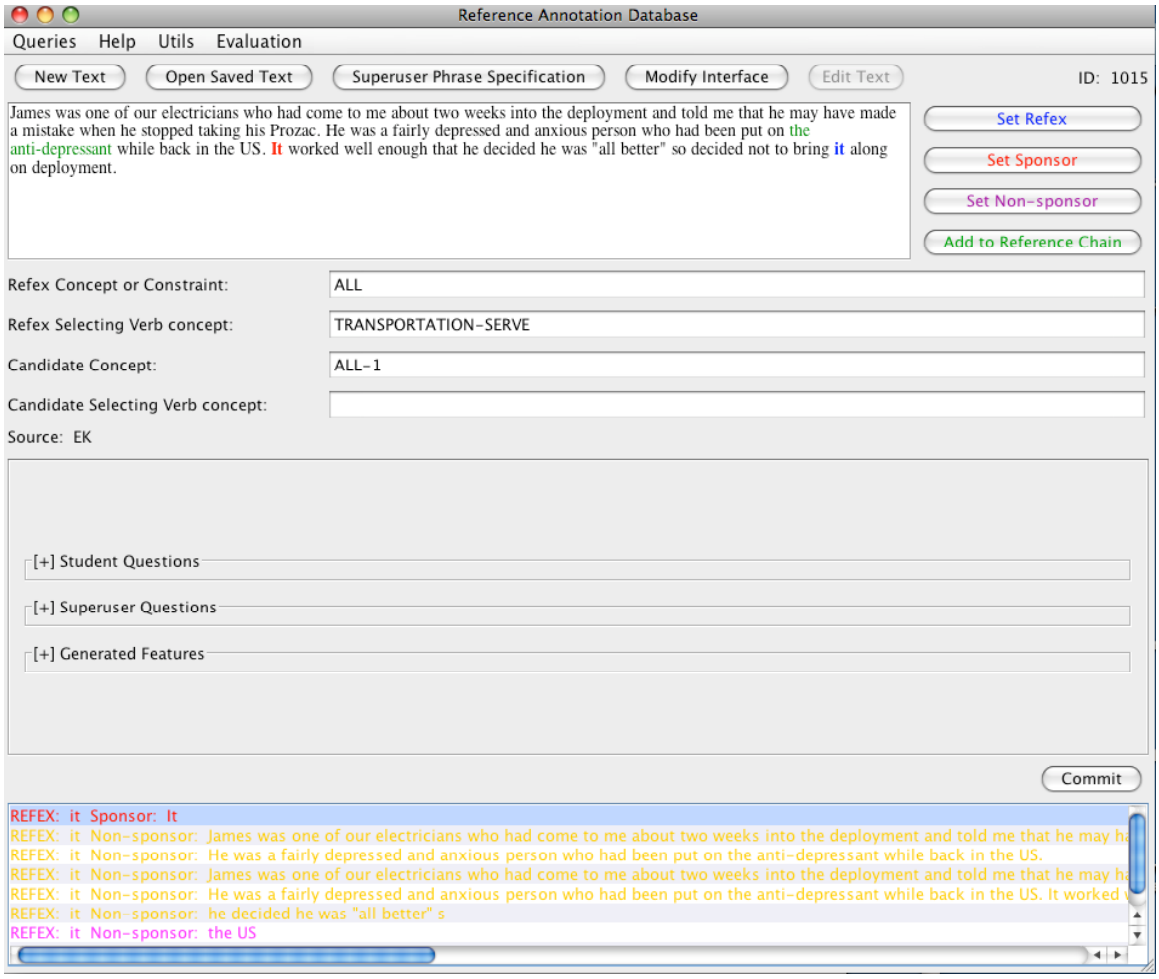

Figure 1. The main screen of the annotation interface.

8 Pedagogical Materials in the Interface

The interface provides an extensive inventory of pedagogical materials describing and providing examples for each linguistic phenomenon elicited. This information is useful not only for training students but for making the environment understandable to and useful for people outside of our group. The style and approach mirror those developed for the Boas knowledge elicitation system (see, e.g., McShane et al. 2003a,b), which was also oriented toward non-specialists who were asked to provide linguistic information about their native language.

The pedagogical materials can be accessed in two ways: by rolling over an elicitation question or by using a help menu. An example of the help menu is shown in Figure 2. If one were to pull up, e.g., the Modality > Overview page, the material in Figure 3 would be presented.

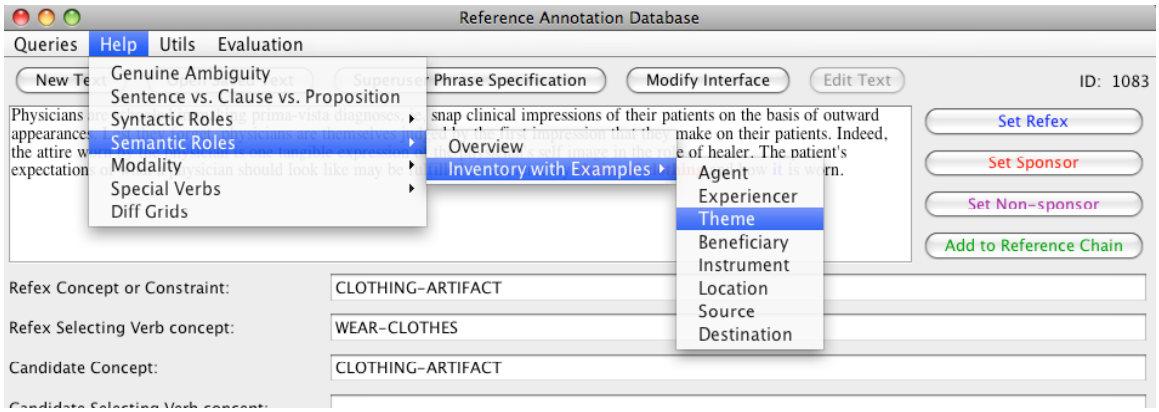

Figure 2. The help menu.

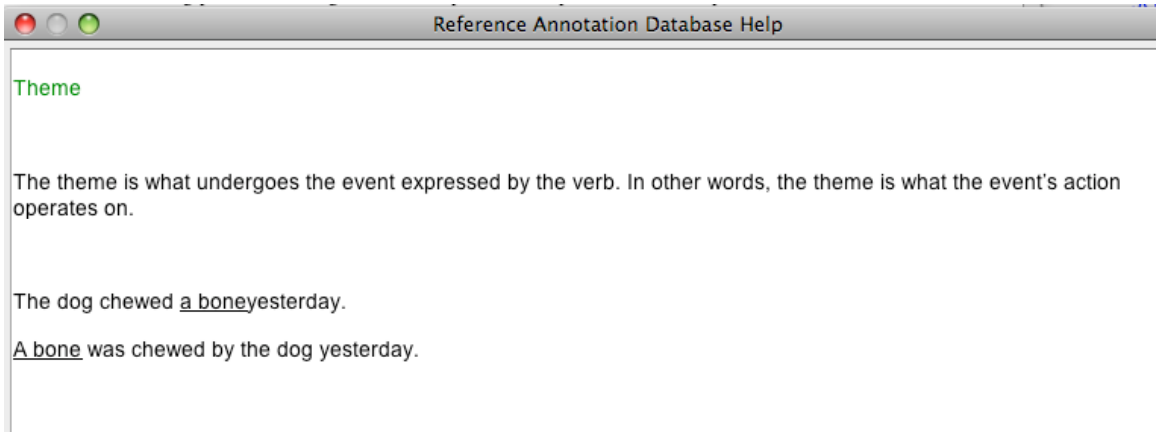

Figure 3. The help page describing the case role “theme”themes.

Rolling over the “RE is instrument” feature value makes the pop-up in Figure 4 appear. The cursor actually points back to instrument in the interface – it simply gets removed when making a screen shot.

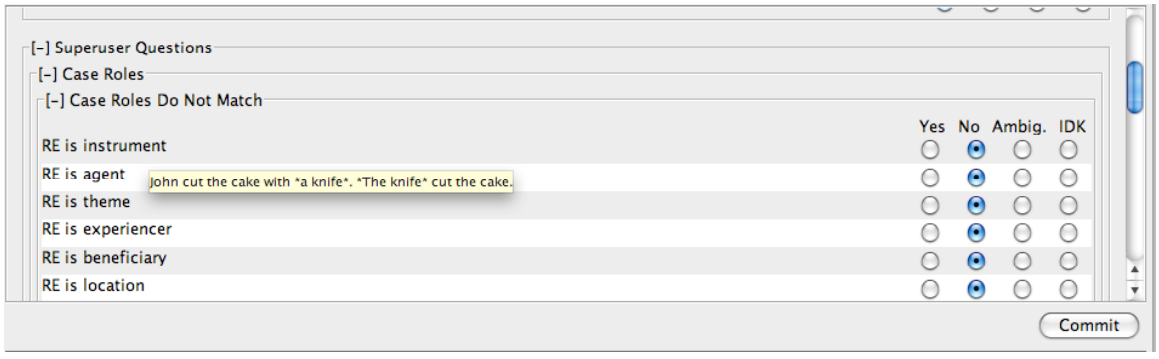

Figure 4. Pop-ups provide examples of features. In this case,”instrument” is shown.

Descriptions from the tutorial materials are provided in part below in the discussions of the various information elicited.

9 Annotation by Lightly Trained Students

Students provided values for a series of conceptually simple properties of all NP candidates. Lightly trained students selected the referring expression (RE), the sponsor, and three non-sponsor NPs. They also provide information about gender, number, syntactic function, various aspects of participation in a chain of reference, if applicable, and a few other bits of information.

The elicitation of gender, number and syntactic function – as well as semantic role, as discussed below – is organized hierarchically. The high-level question in each category asks if the candidate matches the RE's gender/number/syntactic function/semantic function, and the follow up questions seek an indication of exactly what the gender/number/syntactic function/semantic function of the RE and the candidate are. The reason for this two-level classification is to provide options for the creation of predictive hypotheses for reference resolution. It is possible, for example, that a feature value combination (i.e., predictive combination of feature values) will include the property values "genders match" (generalized) and "numbers match" (generalized), but "the candidate is a subject" (a more specific feature value). Only experimentation will tell which levels of which kinds of properties provide the best predictive power.

There are two special questions in this section relating to syntactic function: i) Is the candidate the object of a preposition (known to be a reference-reducing value)? and ii) Are there other NP(s) in the same syntactic position intervening between the candidate an the RE (another reference-reducing value)?

Of course, many other aspects of syntax can be used for reference resolution, select ones of which are elicited in an annotation task carried out by more highly trained students.

Inputting context, selecting RE, sponsor and non-sponsor candidates

The first annotation task, for minimally trained students, is inputting each four-sentence context (or < four sentence context, if all four sentences are not available) and selecting the RE of interest, the actual sponsor, and three NP candidates that are non-sponsors. According to our definition of "context", each context has only one RE of interest. If the same 4-sentence string contains more than one RE of interest, then it is saved as two "contexts", each with a different RE of interest.

The reason for selecting only 3 NP candidates is expense: each candidate needs to have its properties supplied manually, and the more candidates, the longer it takes to annotate each context. The guidelines for selecting non-sponsors were:

(a) do not include humans, since in the vast majority of cases they cannot be the sponsor for *it/that/this* (contexts like *That is my brother* is an exception, but the fact that it requires a human sponsor is easily detected automatically so we are not including such cases in this project)
(b) select sponsors starting from the closest to the RE and working backward

(c) do not select components of compound NPs separately (select *brick wall*, not also *brick* and *wall*)
(d) do include NPs in relative clauses (in *the brick that John threw flew through the window,* select *the brick* and *the window*)
(e) do not spend an undue amount of time choosing the non-sponsors: our goal is to give the system a reasonably difficult choice space without turning non-sponsor selection into a guidelines-ridden enterprise.

Note that even if the actual sponsor is a text span, the annotator still selects three NP non-sponsors because the system does not know from the outset whether the sponsor is an NP or a text span. There is no need for the annotator to select text-span non-sponsors manually because they are generated automatically from the information provided by the superuser in a different annotation subtask. That subtask also provides the few features needed about them.

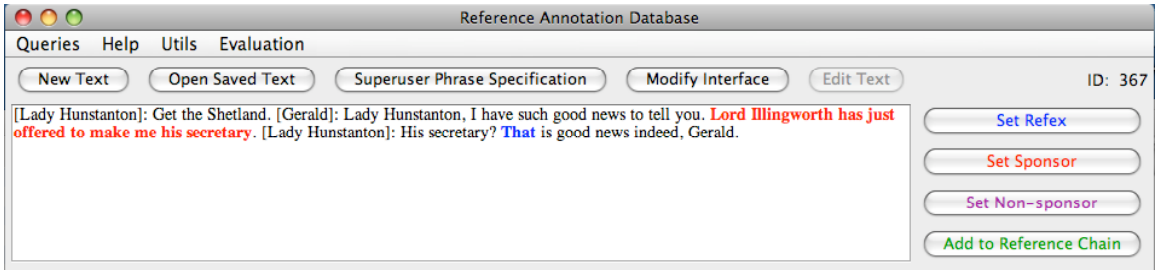

Figure 5. Highlighting of the RE and sponsor. Non-sponsors are highlighted in purple and reference chains in green.

Selection of the RE, sponsor and non-sponsors is done using highlighting: the RE is highlighted in blue, the actual sponsor in red, non-sponsors in purple, and chains of reference, if any, in green. A chain of reference occurs when the most proximate sponsor of the RE has its own sponsor in the context. We mark chains of reference in order not to have each member be a separate candidate, thus requiring, in a sense, more than one sponsor to be selected. Only the actual sponsor and the RE are shown in Figure 5 because each candidate – the sponsor and the 3 non-sponsors – must have its properties described separately and, as such, the knowledge is elicited in 4 "layers": all the information about the actual sponsor, and all the information about each of the non-sponsors 1-3. The layers can be flipped through by clicking on the instances of the RE at the bottom of the main interface window (see Fig. 1). So the continuing knowledge elicitation for the example in Figure 1 will be the property values for the actual sponsor. For discussion of the role of chains of reference, see below.

When inputting the text, the annotator corrects any misrepresented characters, indicates speakers in dialogs formatted like plays, and specially marks headings. Currently, we do not use headings as features in our work (though we should, based on work by Ruslan Mitkov on technical texts), but we do use speaker changes extensively.

The direct questions asked of students are as follows. Figure 6 shows the relevant portion of the main elicitation interface with the tree not expanded. Figure 7 shows the

relevant portion of the Modify Interface interface with the tree expanded. (Note: in this document, screen shots of portions of the tree are typically made from the "Modify Interface" interface, which shows the tree such that more of it can be seen at once). Grayed out properties in Figure 7 are inactive: for organizational reasons we have not been completely deleting things from the DB but, rather, hiding them if not currently used; they do not show up in the interface.

Figure 6. The portion of the main elicitation interface used for yes/no questions for lightly trained students.

Syntax

The basic syntactic questions asked of students are shown in Figure 7 and should be self-explanatory.

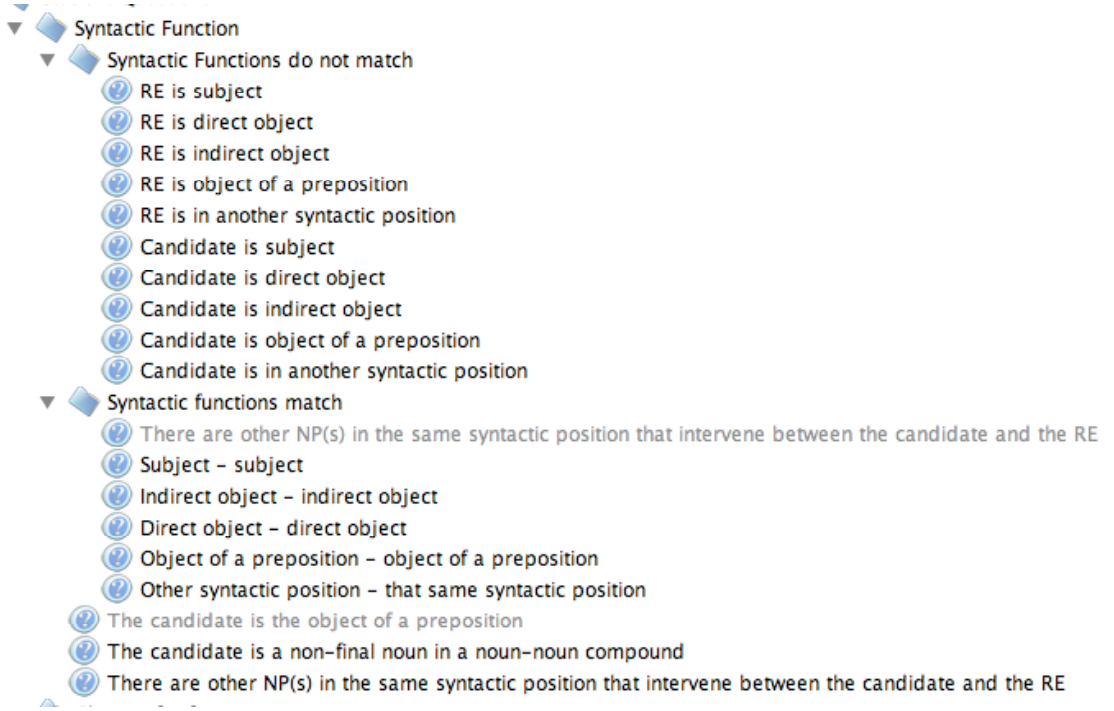

Figure 7. The Syntactic Function question tree shown in the Modify Interface tree.

Chains of Reference

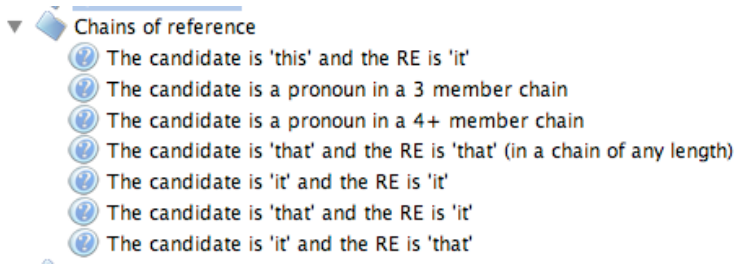

Figure 8. The chain of reference questions, shown in the Modify Interface interface.

It is well-known from the discourse literature, as well as work done on automatic reference resolution, that the longer a chain of reference, the more likely a given RE will be coreferential with it. As such, the annotator indicates whether the candidate itself participates in a 3 or 4+ member chain of reference. By treating chains of reference specially, we avoid the necessity of treating members of a chain as separate candidates, making the selection of a single candidate trickier, at least if one orients heuristics towards things other than text distance between the RE and the candidate. In addition, the annotator indicates if the candidate and RE fall into any of the following classes:

1. the candidate is 'that' and the RE is 'that'
2. the candiate is 'it' and the RE is 'it'
3. the candidate is 'that' an the RE is 'it'
4. the candidate is 'it' and the RE is 'that'

Gender

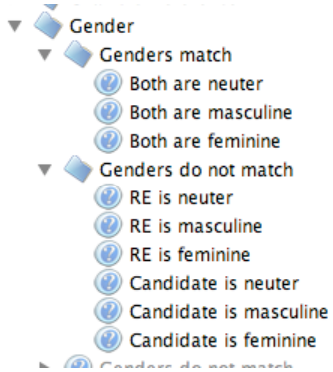

Figure 9. The gender questions shown in the Modify Interface interface.

The only gender used in this project was neuter, since annotators were told not to select non-sponsors that were animate: this would have made the reference resolution task unrealistically easy because most practical systems do quite well at recognizing gender and disallowing gender mismatches in reference. (Of course, there can be gender mismatches in extended usage -- *My computer's great, I just love her* – but we are not pursuing this at this time.) Gender was included in the interface, however, because this

environment will soon be used for annotation of other kinds of REs, like *he, she, they* and definite descriptions.

Number

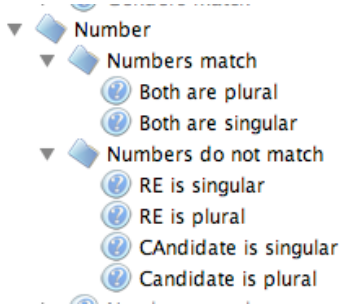

Figure 10. Number questions shown in the Modify Interface interface.

Number is much like gender both in its status in reference resolution overall and in its status in this project. As with gender, annotators were told to try to avoid selecting non-sponsors that were plural because this would make the resolution task too simple. Also as with gender, we know that numbers need not always match for reference, particularly in cases where reference relations are not identity but, for example, a set/instance relationship: e.g., *Mary has two ponies and I want **one** too!* In-depth number issues were not dealt with in this project but will be returned to when we treat plural references and definite descriptions.

10 Annotation by More Highly Trained Students

Case Roles

More highly trained students are asked to assign case roles to all REs and candidate sponsors that are NPs. The inventory is as shown in Figure 11.

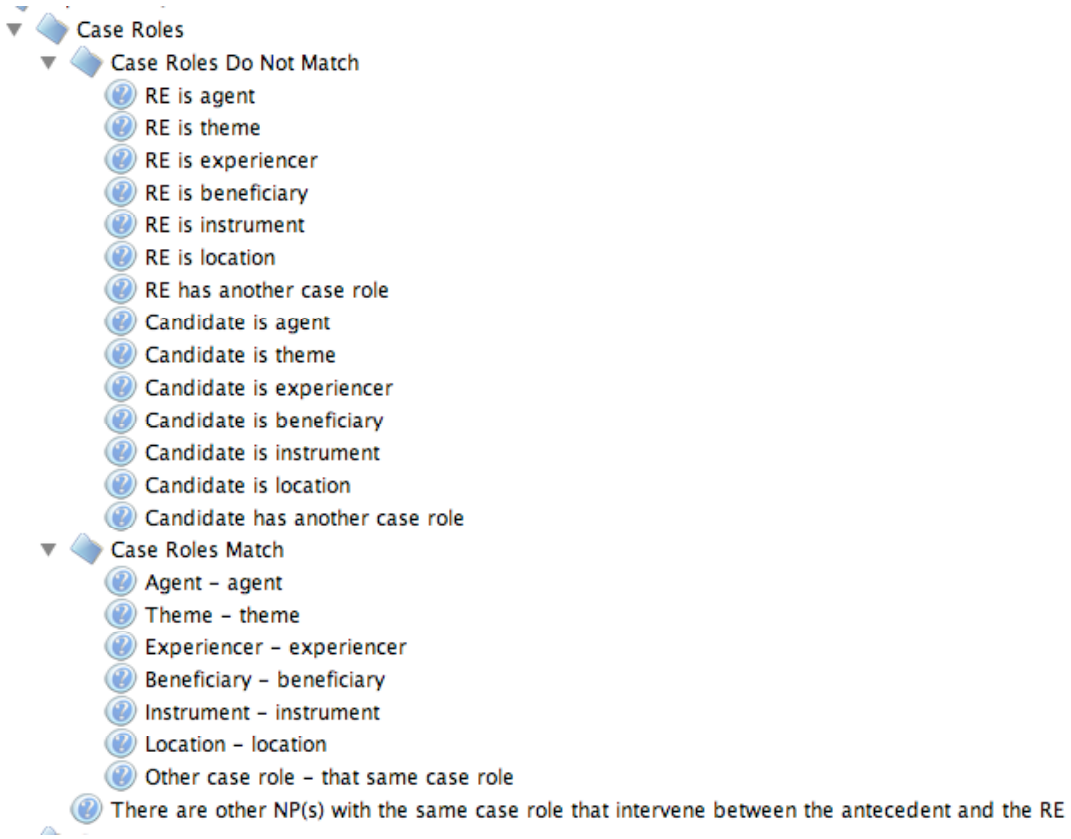

Figure 11. Case role questions shown in the Modify Interface interface.

Constructions

Our initial study of the corpus revealed that many instances of *it/this/that* occur in constructions that suggest their sponsors in relatively easily detectable ways – albeit, as with all aspects of reference resolution, most do not have 100% predictive power. The inventory of constructions below will, we think, provide significant predictive power but in a corpus of the size we used the number of examples found was not large enough to provide a clear picture of widespread usage patterns. This is a case in which linguistic intuitions will need to be used to supply confidence values. For example, in a syntactic construction like *Philosophy: I like it*, the likelihood that the NP before the colon is the sponsor for *it* would seem to be practically 100%. At a later stage of the work, we can try to automatically find examples that have such a structure and create a specialized corpus of them for further study.

Figure 12. Constructions questions shown in the Modify Interface interface.

Targeted Parsing

As shown above, some aspects of syntax are described using questions. Other aspects are described through a highlighting task – carried out by a more highly trained student – that provides information that is later converted into features for reference resolution. These features are used differently for NP candidates than for text-span candidates, the selection of text-span sponsors being more reliant on syntactic characteristics than the selection of NP candidates, for which there are more features of other kinds to rely on.

Our corpus analysis has shown that, when the sponsor for an RE is a text span, clause boundaries and VP boundaries can help to predict what the text span will be. Clause boundaries and VPs are marked only for the text preceding the RE in the RE's sentence (called S0) and for all of the sentence preceding the RE's sentence (S1). The internal structure of S2 and S3 (the 2nd and 1st sentences in the context) are not provided. VPs are marked only in 2 cases:

a. when the RE itself is in the latter part of a VP conjunction structure, in which case an NP sponsor might be in the former part(s): John picked up the ball and threw it. The conjunction might be important, esp. coordinating vs. subordinating
b. when the RE is in the first clause of its sentence and S1 ends with a conjoined VP, in which case the latter conjunct might be the sponsor, especially if it is preceded by a contrastive conjunction: John wanted to buy a car but **didn't have the money**. That's why he didn't buy it.

The clause and VP marking task runs as follows. The highly trained student is presented with an interface that presents the context twice, as shown in Figure 13.

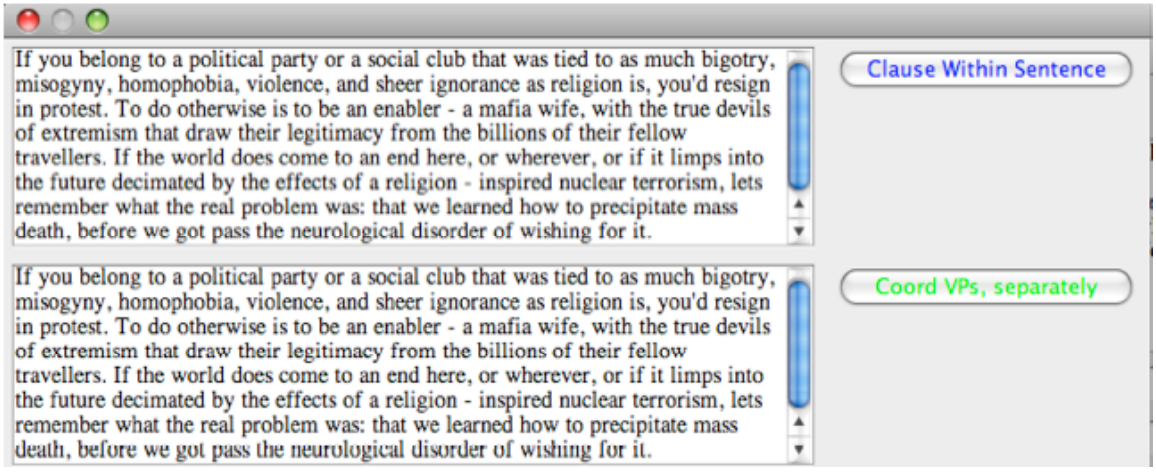

Figure 13. The interface for marking clauses and VPs.

Each copy is marked for one type of entity using a different color. The different copies are to avoid color overlaps when reviewing the annotations.

11 Annotation by Superusers

The only superuser task is to provide ontological mappings for select text strings. These represent the meaning of the text strings after disambiguation has been carried out. These meanings are provided by the superuser with reference to the OntoSem ontology, using the OntoSem lexicon as a time-saving guide for seeking mappings.

For resolving reference, the four most core aspects of context meaning that can be leveraged are: the meaning of the RE, the meaning of each candidate, and the meaning of the selecting verbs for each of these.

The most obvious way to exploit semantics in reference resolution – apart from using case-roles as features, which some systems already do – is to match the semantics of the RE with the semantics of the candidate sponsors. This works well when the RE and the candidates are semantically full. For example, in (1),

(1) When John gave his puppy some food from his plate, the dog wagged its tail with delight.

the dog (instantiated, say, as DOG-2) can readily be coreferred with *the puppy* (DOG-1 (AGE < 1 (MEASURED-IN YEAR))). The candidates *food* (INGESTIBLE-1), *John* (HUMAN-1) and *dinner plate* (PLATE-1) will be rejected since they are ontologically much more distant.

When a RE is not semantically full but has some obvious constraining features, like gender or number (*he, they*), the search space for candidates is also narrowed, albeit less so.

However, the REs *it/this/that* can refer to practically anything, from inanimate objects to propositions and even to people in some cases: *This is my cousin.* As such, their lexically recorded ontological mapping has to be ALL, the base node of the ontological tree of inheritance.

When resolving these REs, a text analyzer can still exploit selectional restrictions but in a different way. The meaning of the REs can be constrained by the semantic expectations of the EVENT that selects them as a case-role. Take, for example, the following context for which the system is trying to resolve the underlined instance of *it.*

(2) "Quite so. It is the problem of slavery. And we are trying to solve **it** by amusing the slaves." (Oscar Wilde)

The word *solve* has only one sense in our lexicon, which is mapped to the ontological concept SOLVE. The immediately relevant properties of this concept are as follows:

```
SOLVE
  IS-A      VALUE   DETERMINE
  AGENT     SEM     ANIMAL
  THEME     SEM     PROBLEM
  …
```

The fact that SOLVE takes an AGENT that is constrained to ANIMAL (dolphins can solve problems too) gives a vote for the fact that this ontological concept might, in fact, appropriately represent the meaning of the context. (Remember, our lexicon and ontology are incomplete, and *solve* might have had a non-agentive meaning not yet recorded.) But the more important fact is that SOLVE takes a THEME constrained to PROBLEM. The noun *problem* is lexically described as mapping to the concept PROBLEM, making the selection of *problem/*PROBLEM-1 quite straightforward.

While one might think that the meaning of the REs *this, that, it* is always highly unspecified, this is not actually the case if one assumes some amount of semantic processing of the context – a kind that is entirely within the capabilities of OntoSem, for example. There are at least two common situations in which the context constrains the meaning of these REs in ways readily detectable automatically:

1. *it/this/that* is the subject of a copular construction in which the predicate nominal or adjective indicates or constrains its meaning: e.g., *That is tractor* [*that* must

refer to an instance of the concept TRACTOR], *It is red* [*it* must refer to an instance of the concept PHYSICAL-OBJECT, since only PHYSICAL-OBJECTs have color, at least in their direct meanings (we discuss non-literal language use ARTICLE but do not cover it in depth in this project)]

2. *it/this/that* is selected by a verb with full semantic meaning whose ontological EVENT mapping includes constraints on the case-role filled by the meaning of *it/this/that*: e.g., in *He ate it, it* must be an instance of INGESTIBLE since *ate* maps to INGEST whose THEME is restricted to INGESTIBLE.

Returning to annotation task, if the meaning of the RE can be determined by method 1, the annotator explicitly enters the ontological constraint on *this/that/this* in the slot for Refex concept and puts X [i.e., not applicable] as the meaning of refex selecting verb concept. If the meaning of the RE can be determined by method 2, then the meaning of the selecting verb is listed in refex selecting verb concept, and the selectional constraints on the appropriate case-role of that concept are listed as Refex concept. Importantly, if the entity is constrained to certain classes of ontological OBJECTs, then only NP candidates need be sought, not text spans. For example, INGESTIBLEs can only be realized as NPs, whereas IDEAs can be realized by spans of text as well. Since one of the most difficult aspects of resolving *it/this/that* to begin with is determining what kind of sponsor should be sought, constraints that rule out an entire class of sponsor types narrow the resolution search space considerably.

We have just motivated providing the meaning of the RE and its selecting verb. The need to provide the meaning of each candidate should be clear, since semantic matching between candidate and RE is arguably *the* most important heuristic for reference resolution. The need to provide the meaning of the candidate's selecting verb, however, requires motivation.

It has been shown that parallelism is key to many types of reference relations. For example,

- McShane 2005 and McShane 2009c show that argument ellipsis in Russian and Polish is highly promoted in configurations that show syntactic and lexico-semantic parallelism
- Lobeck 1995, following the extensive literature on gapping, shows that the verbal ellipsis phenomenon called gapping (e.g., *John prepared his lecture in 3 hours and Mary [e] in 4 hours*) requires syntactic and semantic parallelism of the entities surrounding and licensing the gap
- Carbonnell and Brown 1988 posit a semantic alignment preference that should be used for reasoning about reference in contexts like *Mary drove from the park to the club. Peter went there too*; barring other information, the assumption should be that all departures are from the same place and all arrivals are to the same place.

Correspondingly, in some types of contexts, parallelism can strongly predict the sponsor for *this/that/it*. For example, if (a) the candidate and the RE are in sequential clauses (possibly, also, not sequential clauses, with a lower confidence on the outcome), (b) their selecting verbs map to the same or closely ontologically subsumptive concepts, and (c) if

the candidate and the RE occupy the same respective case-roles, then their coreference is strongly suggested. The type of configuration in question is shown in (3), in which *hit* and *smash* map to the same concept – HIT – and the candidate and RE are both THEMEs of their respective HIT events.

(3) John [hit]$_{HIT}$ [the ball]$_{Candidate/THEME}$, he [smashed]$_{HIT}$ [it]$_{RE/THEME}$ really hard

To leverage this generalization, we must make reference to the relationship between the meanings of the selecting verbs of the RE and each of the candidates. The relationships we are most interested in are shown in Figure 14. They are generated from the actual ontological mappings provided by superusers.

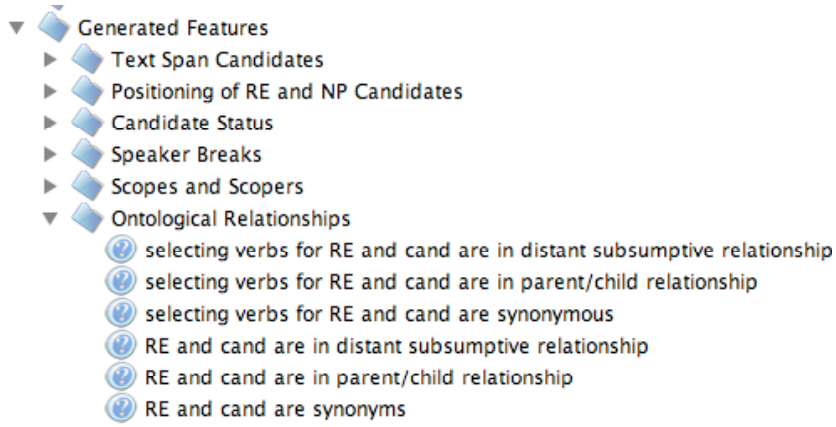

Figure 14. Parallelism-oriented properties.

The expectation is that reference relationships are supported most by synonymy and, in decreasing order, by a parent/child relationship, distant subsumption and none of the above.

Indicating ontological mappings is the most time-consuming annotation task and must be carried out by a superuser. Given more development time, it could have been semi-automated by having the annotator highlight the selecting verbs (RE, sponsor and non-sponsor candidates have already been selected by the student) then have the system search for all the ontological mappings of each relevant word in our lexicon and present them to the annotator as a crib.

Due to the time factor, several simplifications were made to the task of supplying ontological mappings.

- If the selecting verb is 'be', then it is marked as X, since 'be' does not have an ontological mapping in the theory of Ontological Semantics. (E.g., "The car is red" is the proposition (CAR (COLOR: RED)).
- If the selecting verbs for the RE and the candidate clearly have no close relationship that would enhance predictive power, no mappings were provided.

- If any of the words that requires an ontological mapping was not in the OntoSem lexicon, then a coarse-grained mapping was supplied.

During the course of testing, we encountered a number of unexpected situations that led to further modification of the task of supplying ontological concepts.

Recall that the strongest use of semantics for reference resolution is when the mappings are highly constrained: e.g., the candidate means DOG, or the selecting verbs for the candidate and RE both map to INGEST. If, by contrast, the mappings are very high on the ontological tree – such as OBJECT or PHYSICAL-EVENT, then they provide little predictive power. Of course, in an actual system they would be essential for ruling out candidates that are of the wrong ontological type; but for our current purposes, they led to an unexpected problem.

We expected the feature value "RE and candidate are synonymous" and "RE and candidate have the same selecting verbs" to be very strong predictive features, since we were envisioning cases like DOG ~ DOG and INGEST ~ INGEST. However, in our first test of this feature many candidates that showed such synonymy with the RE were actually not the sponsors. The reason for this is that the matching concepts were either X (the existential verb) or they were very high on the ontological tree (to reiterate, the latter typically occurred because the word was initially missing from the lexicon and the user mapped it coarsely). Clearly, these kinds of matches were not what we had in mind for these features. Therefore, we made the following reinterpretation of the concept mappings:

- If the RE's selecting verb is existential, its value should be X, but if the candidate's selecting verb is existential, its value should be X-1. This means that the automatic process that generates the feature values "RE and candidate selecting verbs are synonymous" will NOT consider these synonymous but we can still find the examples of this type readily using a DB search, if needed.
- Similarly, if the RE maps to an ontologically high level concept, that concept is written in directly, but if the candidate maps to a high-level concept, that concept has a -1 appended to it. Again, synonymy will not be detected.

The main idea is that this annotation effort is focusing on ontologically constrained synonymy. Ontologically less constrained synonymy is useful in a system like OntoSem, which uses semantics in far more ways than are reflected in this annotation effort. For future work, we need to create a convention to indicate that the selecting verbs have similar semantic functions even if they do not have an ontological mapping. For example, modal and aspectual verbs do not map to concepts but if they are used in parallel in a text, they can have predictive power: e.g., *He started on **his hike** at 10 a.m. Because he set out on **it** so late, it got very hot when he was only halfway through.*

Some practical notes are as follows. At this time, only one concept or constraint can be put in a given slot in the interface. If the selecting verb's slot is supposed to be filled by constraints found in the ontology, and if there is more than one constraint on the sem facet (the basic semantic constraint) then there is an annotator choice not governed by strict rules:

- If the constraints are very close in the tree (e.g., DOG and CAT) and they are readily subsumed under a single not-too-high concept (e.g., HOUSEPET or whatever we have) then use that ancestor
- If the constraints are really far apart in the ontology (i.e., no useful ancestor), then just choose one.

12 Automatically Generated Features

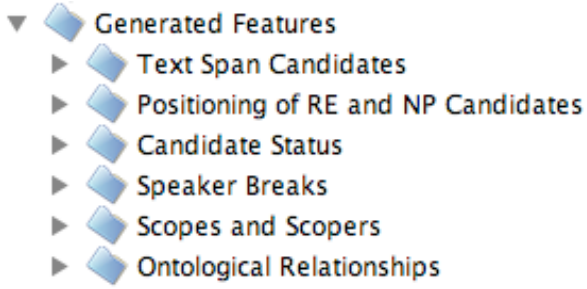

Figure 15. The types of automatically generated features in the system.

We have discussed the relevant aspects of most of what we call "automatically generated features" from the point of view of acquisition, above. Details about text span candidate generation are provided here.

All texts span candidates are generated automatically based on a linguistically informed algorithm. The algorithm, presented below, takes into account a number of considerations.

In order to act as a sponsor for it/that/this, a text span typically must be **contiguous** with the sentence containing this/that/it – a generalization grounded in previous research (see, e.g., Byron 2004 and the references therein). There are exceptions to this, as when expletives, asides, etc., intervene, but we did not pursue these exceptions in the work to date. This generalization about contiguity means that when the system automatically generates all reasonable text spans that the reference resolver must choose from, it will only generate contiguous ones. Indeed, it would be ridiculous to consider as a candidate, for example, the 2nd clause in a 4-clause sentence located 3 sentences back from the RE. If we did this, we could generate hundreds of moot candidates. If a non-contiguous text span actually is the sponsor, the annotator can still explicitly choose it, but it will not match any automatically generated candidates. We are following the working hypothesis that text span candidates can "skip over" previous material only if it is in S0. This hypothesis requires further testing.

In the current implementation, text span candidates can include previous components of S0 or combinations of portions of S1 (with clauses tacked on right to left), but **sentences S2 and S3 are not "split up" into clauses for purposes of text span candidate generation**. Fully annotating S2 and S3 for clause boundaries would have

taken too long and it is not clear to what extent it would have been helpful – i.e., how often will the actual sponsor be, say, the last clause in S3 plus all of S2 and S1?

We also count clauses going backwards. In the following sentence there are 2 clauses preceding the RE clause which, when combined, make a 3rd clause.

[$_{cl3}$ [$_{cl2}$ He is tall] and [$_{cl1}$ he is handsome]] and <u>that</u> is why she considers him attractive.

The candidate 'he is handsome' is referred to as "text span cand is 1 of 3 cl in S0"
The candidate "he is tall and he is handsome" is referred to as "text span cand is 3 of 3 cl in S0"

We are assuming that there will be a maximum of 4 preceding clauses in S0 and 6 clauses in S1.

All text span candidates are generated automatically based on the following sources of information:

1. sentence boundaries marked by ? or !, which were detected automatically
2. sentence boundaries marked by a period, which were provided manually by annotators; a special, very quick interface was created for this; the need for manual marking derived from a high error rate of automatic preprocessing, since periods are multiply ambiguous
3. clause boundaries in S0 and S1, provided manually by annotators
4. VP boundaries in S0 and S1, provided manually by annotators

Sentence, clause and VP breaks are represented in the databases as string indices, in the same way as referring expressions, sponsors and text spans selected by annotators as actual sponsors. (In the future, two organizational aspects of this process need to be changed: the sentences in a context should be stored individually so that sentence breaking is not a problem, sentences should not be manipulated using indices, and special means for treating indications of speakers, titles, subtitles, etc., must be developed. The latter was an oversight that then required manual intervention for this round of testing and evaluating. Namely, speaker indications were considered part of the sentence by the text span candidate generator, so an annotator's selection of a direct-speech context as a sponsor would never match the system's, since the annotators were not including speaker indications in their spans whereas the system was.)

Following the generalizations and constraints stated above, the automatic generation of text span candidates is carried out according to the following algorithm:

Determine if there are any clauses before the RE's clause in S0. If there are none, then S0 is empty.

If S0 is nonempty:

Determine endIndex for all text spans in S0. The end of all text span candidates that involve S0 is the last character (minus punctuation) of the last word of the clause preceding the RE's clause.

Create text spans that involve only S0 using

- **endIndex**, as calculated above
- **startIndex** provided by (a) annotators carrying out the clause boundary indication task and (b) the rule that the beginning of each sentence is a left-hand clause boundary. (Note: (b) overgenerates, the problem being adverbs and such at the beginning of sentences, esp. when they should not be part of the candidate; see evaluation section for further description)
- **the rule** that text spans must be contiguous to the RE's clause, so if there are multiple clauses in S0, they are concatenated right to left.

E.g., in the context

S3 S2 S1 [$_{cl3}$ He ate breakfast, [$_{cl2}$ walked the dog and [$_{cl1}$ went to work]]], doing all <u>that</u> with his pajamas on!

the text span candidates in S0 are, working backwards

went to work
walked the dog and went to work
He ate breakfast, walked the dog and went to work

(We are avoiding the double-left-hand edge problem at the beginning of the sentence, mentioned above.)

If S1 is non-empty, create text spans that involve only S1 using

- **endIndex**, which is the last non-punctuation character of S1
- **startIndex** provided by (a) annotators carrying out the clause boundary indication task and (b) the rule that the beginning of each sentence is a left-hand clause boundary
- **the rule** that relevant text spans in S1 are all of S1 and, as applicable, the last clause of S1 and any other clause combinations created by concatenating clauses right to left (the last 2 of 3 clauses, the last 3 of 4 clauses, etc.)

If S0 is non-empty and S1 is non-empty, create text spans that involve just S0 and S1

- for each text span candidate in S1, add on the largest text span candidate in S0, creating a new candidate.

If S2 is non-empty, create one text span using

- **endIndex**, which is the last character of S1 (not punctuation)
- **startIndex**, which is the first non-punctuation character in S2.

Recall that the internal structure of S2 is not provided or used.

If S2 and S0 are non-empty, create one text span using

- **endIndex**, which is the last character of the last word in the S0 clause before the RE's clause (not punctuation)
- **startIndex**, which is the first non-punctuation character in S2.

If S3 is non-empty create one text span candidate using

- **endIndex**, which is the last character of S1 (not punctuation)
- **startIndex**, which is the first non-punctuation character in S3.

Recall that the internal structure of S3 is not provided or used.

If S3 and S0 are non-empty, create one text span using

- **endIndex**, which is the last character of the last word in the S0 clause before the RE's clause (not punctuation)
- **startIndex**, which is the first non-punctuation character in S3.

If the RE is in the latter VP of a coordinate VP structure in S0, create one text span using

- **endIndex**, which is the last character of the last word in the VP before the RE's VP
- **startIndex**, which is the first character in the VP preceding the RE's VP

VPs preceding the RE in S0 and in the final position of S1 are indicted explicitly by annotators.

If the RE is in the first clause of S0 and S1 ends with a VP, then create one text span using

- **endIndex**, which is the last character of the last word in S1
- **startIndex**, which is the first character in the VP that ends S1

Some examples:

John ate a burger. He drank a coke. Then he went swimming. **That**'s how he spent his evening.

The candidates, working backwards:
(Then) he went swimming. ; inclusion/exclusion of the adverb is an issue
He drank a coke. Then he went swimming.
John ate a burger. He drank a coke. Then he went swimming.

John ate a burger. He drank a coke and **he nursed a beer.** Then he went swimming and **fishing**. That's how he spent his evening.

fishing.
(Then) he went swimming and fishing.
nursed a beer. Then he went swimming and fishing.
He drank a coke and he nursed a beer. Then he went swimming and fishing.
John ate a burger. He drank a coke and he nursed a beer. Then he went swimming and fishing.

Since it might not seem obvious why VPs should be considered separately, consider the following example, where a VP after *but* is the most likely sponsor (the entire proposition could also be considered the sponsor: this is a case of benign ambiguity).

John ate some of his burger but **didn't finish it**. That made his mother mad.

The static inventory of text span candidates, automatically selected for a context as applicable, is as follows. The candidates in red are generated automatically using the initial inventory of candidates as input.

text span cand is S1
text span cand is S1 & S2
text span cand is S1, S2 & S3

text span cand is 1 of 1 preced cl in S0
text span cand is 1 of 2 preced cl in S0
text span cand is 1 of 3 preced cl in S0
text span cand is 1 of 4 preced cl in S0
text span cand is 2 of 2 preced cl in S0
text span cand is 2 of 3 preced cl in S0
text span cand is 2 of 4 preced cl in S0
text span cand is 3 of 3 preced cl in S0
text span cand is 3 of 4 preced cl in S0
text span cand is 4 of 4 preced cl in S0
text span cand is all preceding text in S0
text span cand is part of preceding text in S0

text span cand is 1 of 2 cl in S1 ; ½, 2/2, etc. covered by "is S1"
text span cand is 1 of 3 cl in S1
text span cand is 1 of 4 cl in S1

text span cand is 1 of 5 cl in S1
text span cand is 1 of 6 cl in S1
text span cand is 2 of 3 cl in S1
text span cand is 2 of 4 cl in S1
text span cand is 2 of 5 cl in S1
text span cand is 2 of 6 cl in S1
text span cand is 3 of 4 cl in S1
text span cand is 3 of 5 cl in S1
text span cand is 3 of 6 cl in S1
text span cand is 4 of 5 cl in S1
text span cand is 4 of 6 cl in S1
text span cand is 5 of 6 cl in S1
text span cand is part of S1

text span cand is all preceding text in S0 & 1 of 2 cl in S1
text span cand is all preceding text in S0 & 1 of 3 cl in S1
text span cand is all preceding text in S0 & 1 of 4 cl in S1
text span cand is all preceding text in S0 & 1 of 5 cl in S1
text span cand is all preceding text in S0 & 1 of 6 cl in S1
text span cand is all preceding text in S0 & 2 of 3 cl in S1
text span cand is all preceding text in S0 & 2 of 4 cl in S1
text span cand is all preceding text in S0 & 2 of 5 cl in S1
text span cand is all preceding text in S0 & 2 of 6 cl in S1
text span cand is all preceding text in S0 & 3 of 4 cl in S1
text span cand is all preceding text in S0 & 3 of 5 cl in S1
text span cand is all preceding text in S0 & 3 of 6 cl in S1
text span cand is all preceding text in S0 & 4 of 5 cl in S1
text span cand is all preceding text in S0 & 4 of 6 cl in S1
text span cand is all preceding text in S0 & 5 of 6 cl in S1
text span cand is all preceding text in S0 & all of S1
text span cand is all preceding text in S0 & part of S1

text span cand is all preceding text in S0 & S1, S2
text span cand is all preceding text in S0 & S1, S2, S3

text span cand is the VP in S0 preceding the RE's VP
text span cand is the sentence-final VP in S1

Below is a shot of the modify interface screen that shows a subset of this inventory of features:

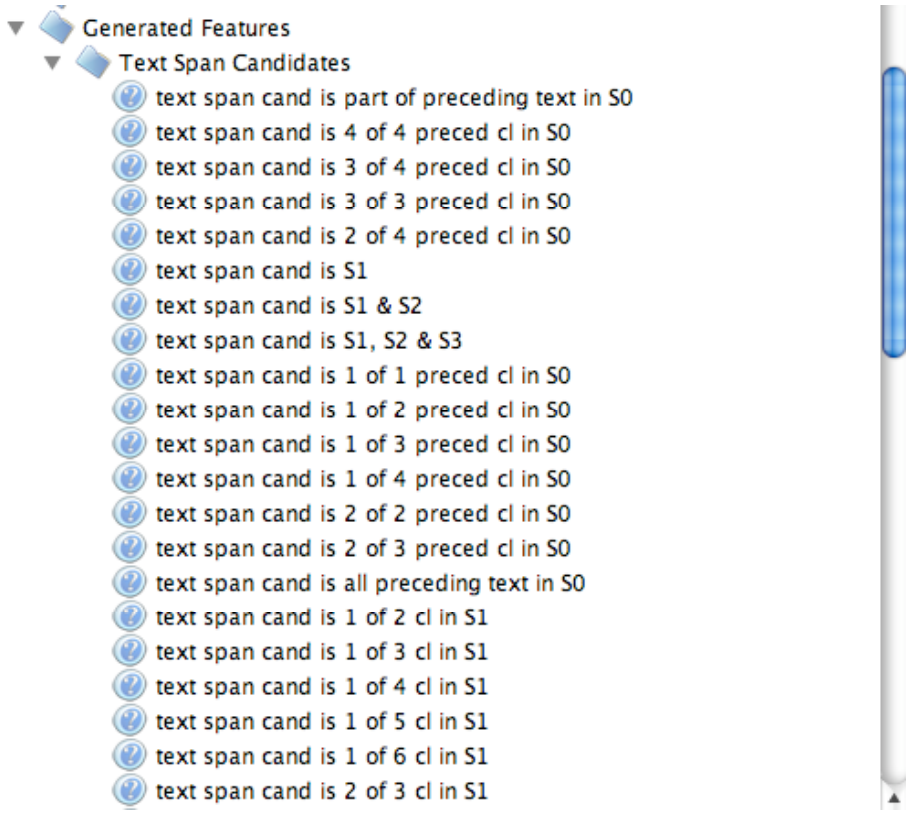

Figure 16. A portion of the inventory of text span candidates as shown in the Modify Interface interface.

Switching gears to another kind of generated feature, the positioning of NP candidates has been found to be a strong heuristic, the closer the candidate (barring candidates that are arguments of the same verb as the RE), the higher the chance it is the actual sponsor. At first we made many subtypes of these features such as "the candidate is 2 clauses away"; however, these distinctions became far too numerous and lacked generalizing power. We still have the information to create more features, such as "2 clauses away" from the data we have already input, we simply have not promoted them to features at this time.

RE is in first clause of S0
NP cand is in S0
NP cand is in S1
NP cand is in S2
NP cand is in S3
NP cand is in the VP conjoined with the RE's VP
NP cand is closest of the NP candidates
NP cand is 2nd closest of the NP candidates
NP cand is 3rd closest of the NP candidates
NP cand is farthest away of the NP candidates

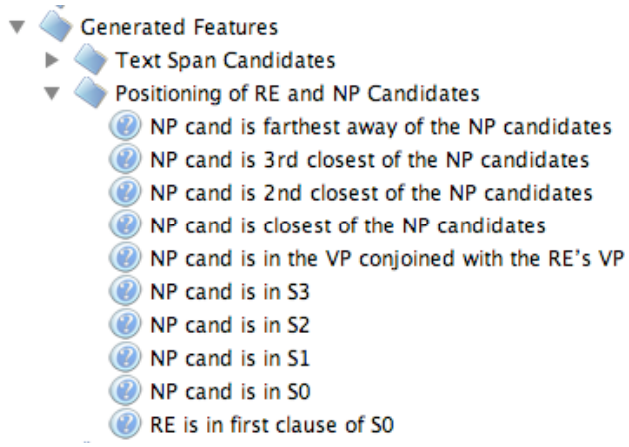

Figure 17. The positioning features as shown in the Modify Interface interface.

13 Human-Machine Collaboration in Creating Feature Value Combinations and Their Confidence Levels

Combinations of feature values that provide some power to predict whether a candidate is a sponsor could, in principle, be created in many ways. For example,

1) Human only: People can think up features and feature combinations and assign them a confidence level using exclusively introspection.

2) Machine only, following manual annotation: An engine could create all combinations of feature values then run them over the examples of an annotated corpus, returning feature value combinations and associated confidence levels. Depending on the query sent to the engine, feature value combinations and their associated weights might be returned as follows:

When the following feature value combinations appear on a candidate, it is always the sponsor

- X, Y, Z
- A, B, C, D, E
- F, G

When the following feature value combinations appear on a candidate, it is the sponsor in all cases but 1 (but 2, etc.)

- M, N, O, P
- Q

When the following feature value combinations appear on a candidate, it is never the sponsor

- I
- J, K

3) Human-machine collaboration: People can:

a. study a corpus
b. think up features and their value sets that might contribute to selecting a sponsor
c. annotate a corpus for these features
d. think up combinations of those features that they believe might have high predictive power based on the examples in that corpus
e. automatically determine how frequently each feature value combination is found on sponsors and how often on non-sponsors
f. refine the inventory of feature values in given combinations, if possible, to give them more predictive power without the loss of too much coverage
g. assign a confidence value based on the results of steps (e) and (f) – and, optionally, with the inclusion of introspection if the corpus is not large enough to provide sufficient examples of all Blocs. To take an intentionally extreme example, if in our corpus a candidate that is syntactically a subject and semantically a theme were always the correct sponsor, this generalization should not be incorporated into our reference resolution algorithm as a strongly predictive rule because we can readily invent realistic counterexamples. On the other hand, if we have a generalization that seems linguistically strong but we do not have enough examples to strongly support its generality, we might want to include it in our algorithm anyway, at least until proven wrong by more corpus evidence.
h. configure a reference resolution engine that uses the weighted feature value combinations and run it on another corpus as evaluation.

We used method #3. Although we considered using method 2 – and, indeed, put quite bit of work into carrying it out – we did not complete that work due to the problem of combinatorial explosion (we have a lot of features) and the difficulty in constraining the problem space to circumvent that. If we are, in the future, to use a "more automatic" method of creating feature value combinations, we will first need to reduce the search space very significantly using linguistically-motivated generalizations. To take a very simple example: only one "level" of a given feature should be used at a time (recall we have a hierarchy of features for certain kinds of features). For example, if one uses the highest-level generalization about gender –that the genders match or don't match – then the more specific gender features must be ignored, such as "both are masculine" and "the RE is masculine". This is something that would be done as a matter of course in manually creating feature value combinations, but would not be automatically done if the search space for a fully automatic process were not constrained. Another problem with using method 2 is that our corpus is rather small, meaning that the generalizations we were getting on early testing were not very representative of what we would expect over a large data set.

14 The Creation, Testing, Scoring and Evaluation of Sets of Feature Value Combinations

The workflow for creating groups of feature value combinations is as follows.

Step 1 Create folder of examples

Select an inventory of contexts to work with – a training/experimental set – and save them to a folder (Figures18 & 19).

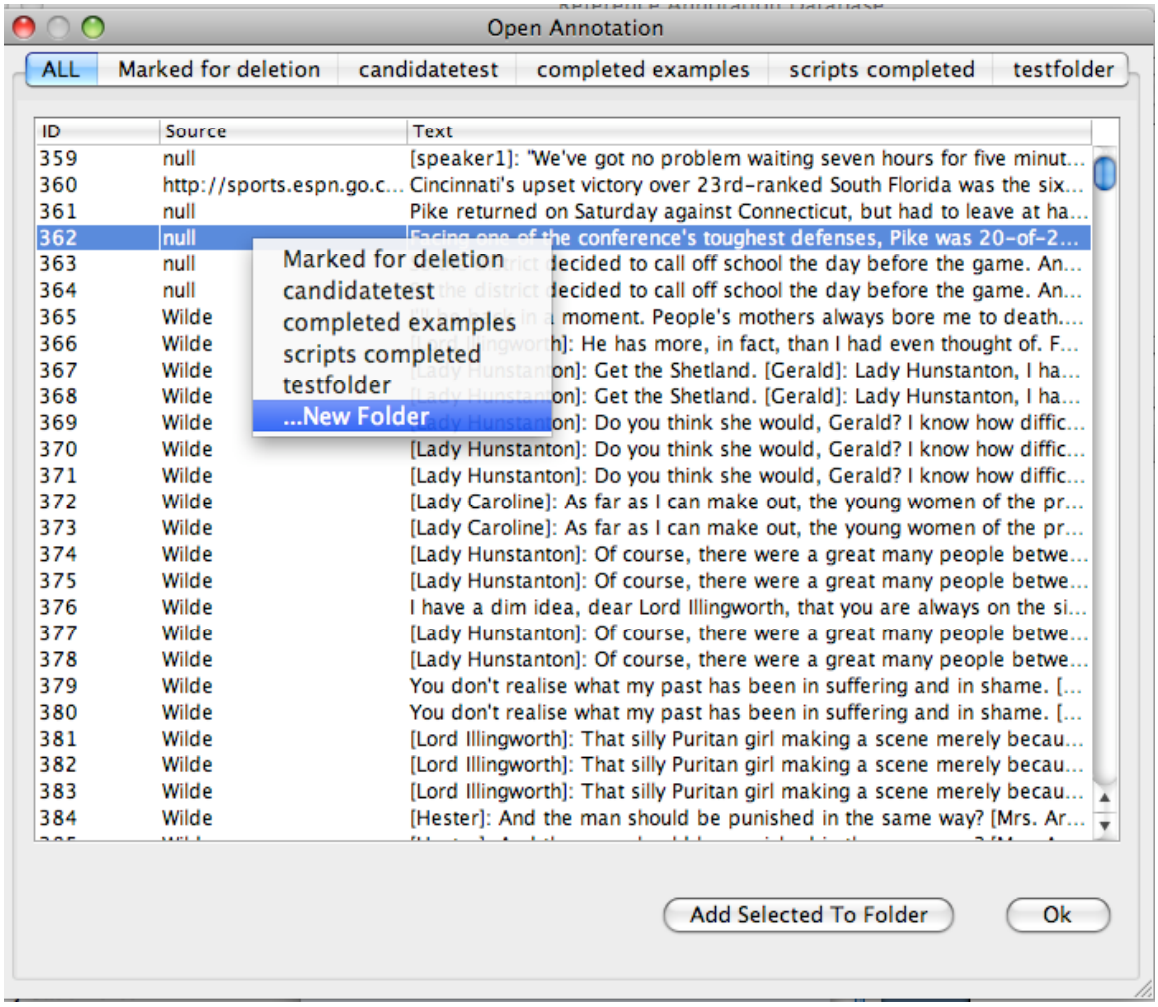

Figure 18. Creating a corpus to work on from the entire inventory of examples.

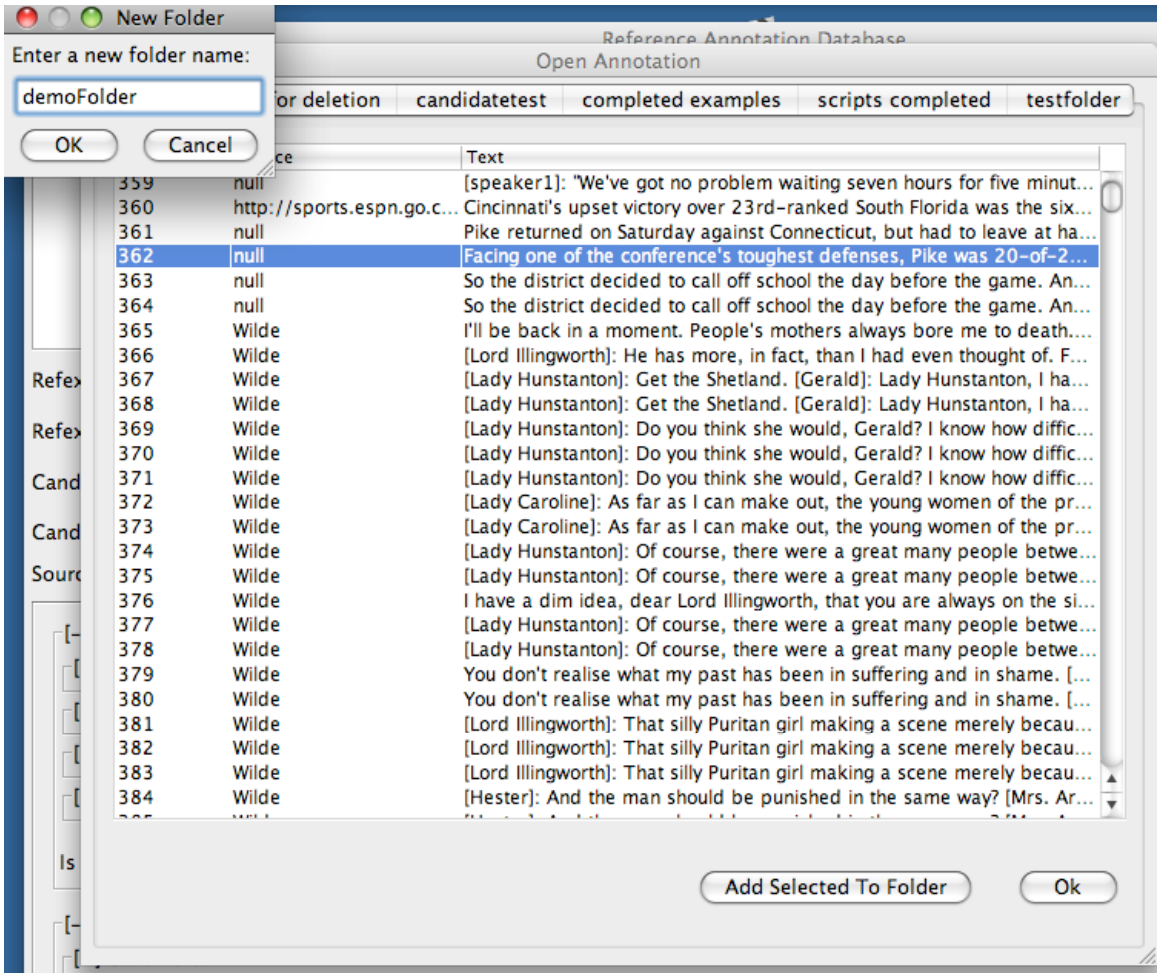

Figure 19. Naming the folder.

2. Create feature value combinations

Create, through introspection, a set of feature values that one thinks will have significant power to predict a sponsor or a non-sponsor. (In some cases, one might want to test combinations or singletons that one believes will not have very high predictive power for comparison.) The available feature values are the same as in the elicitation task: yes, no, I don't know and ambiguous (the latter two are not used at this time). For example, in Figure 20, the following feature value combination is selected:

subject-subject yes ; the candidate and RE are both subjects
theme-theme yes ; the candidate and RE are both themes
NP cand is closest of the NPs candidates yes

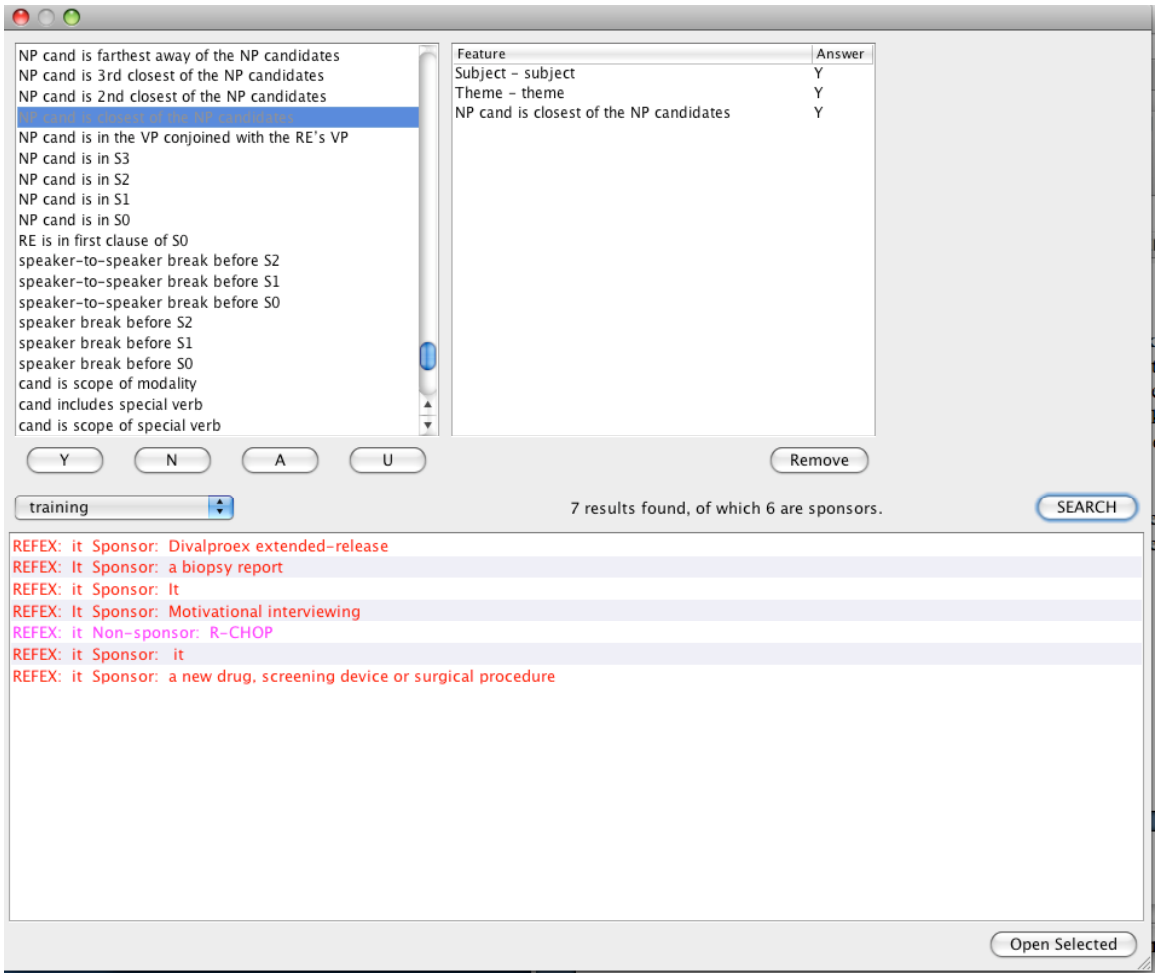

Figure 20. The interface for selecting a feature value combination and seeing how often it appears on a sponsor and how often on a non-sponsor.

The upper left frame contains the list of all the features used in the system, both directly elicited and generated from elicited values. The upper right frame shows the features selected by the user for the given query and their values. The pull-down menu permits the choice of a folder of examples to work on.

Step 3. Test the set of feature values

The Search button in Figure 20 launches a search through all the examples in the selected folder, seeking candidates (sponsors or not) that match the criteria. Actual sponsors are returned in red, non-sponsors in pink. A tally of how many results were found and how many of them were sponsors is shown. The Open Selected button permits the user to open any of the contexts in an attempt to understand why a query had an unexpected result, what other feature values might narrow the query in a useful way, etc. For example, when the last context in Figure 19 is opened in the normal elicitation interface it looks as in Figure 21.

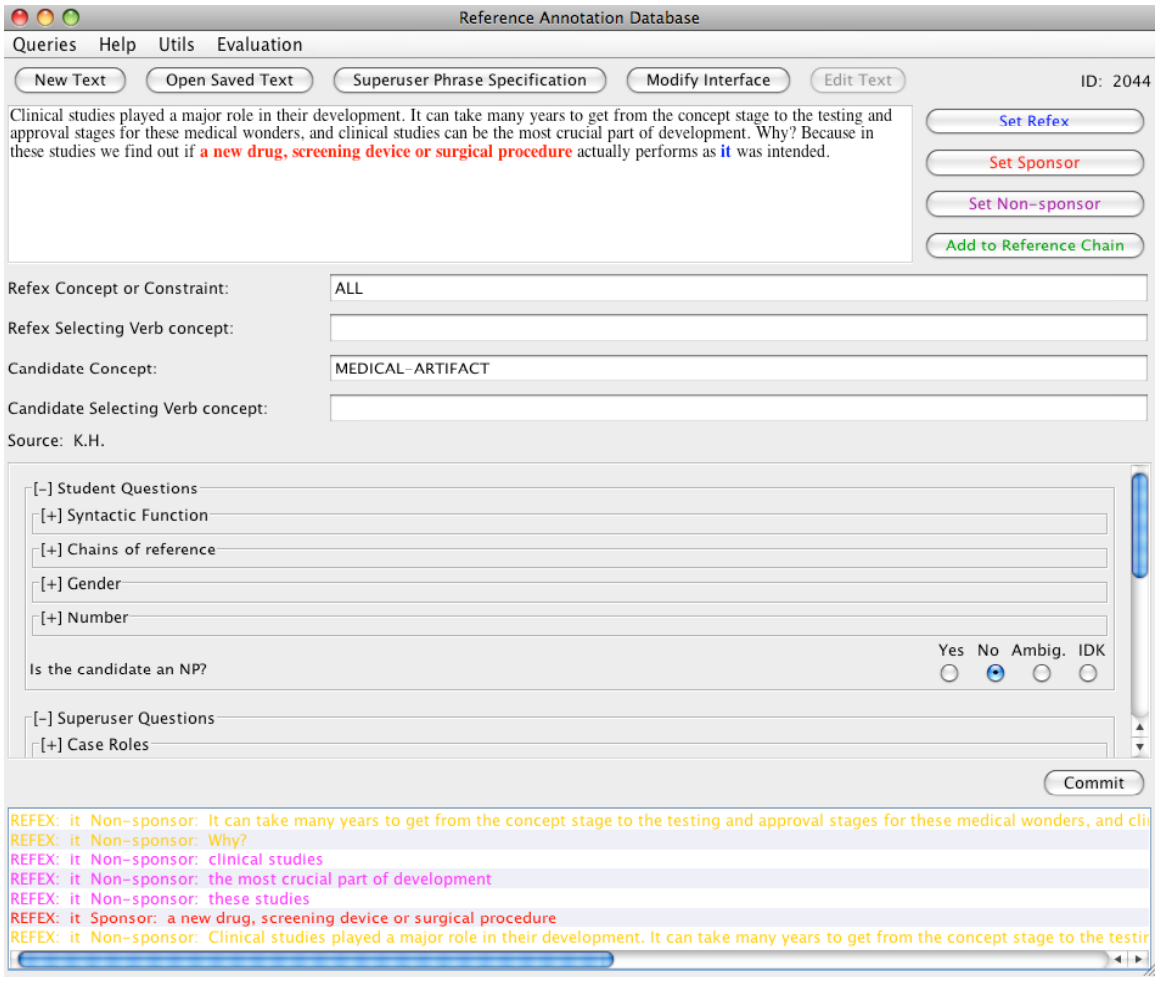

Figure 21. Opening up the last hit from the search shown in Figure 19 in the main elicitation pane.

Here, the data can be reviewed or edited if the evaluation process helped to detect an error. The results of the query in Figure 20 show that this is a reasonably predictive query, finding 6 sponsors of 7 hits, or 86%. This will be a "2nd base hit" in our terminology (see the evaluation section for details). Compare this with the query in Figure 22, whose results are not very predictive at all: 12 of 25 hits are sponsors.

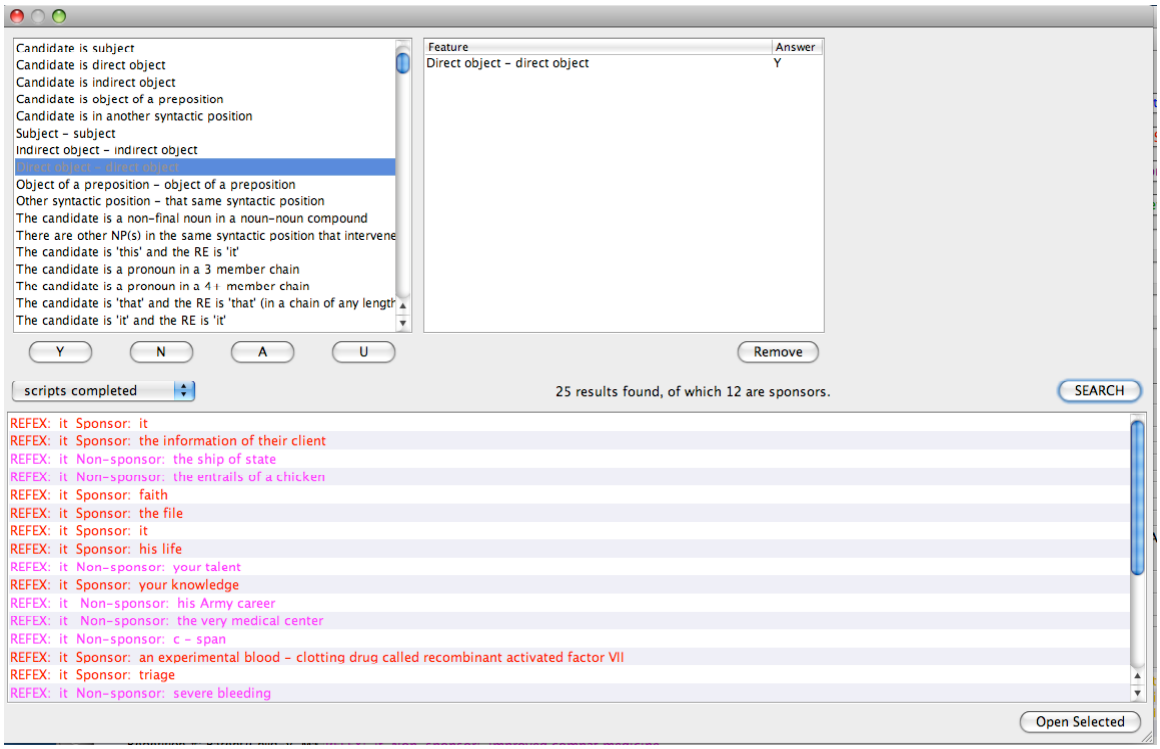

Figure 22. A query that is not very predictive, since this feature value shows up on quite a lot of sponsors and non-sponsors.

4. Study the results of the feature value combination search

The results of the feature value combination can be studied, particularly those that defied our linguistically-based expectations, to determine if any other feature values might make the hypothesis stronger. If feature values can be added for more predictive power, it does not mean that we need to discard the original feature value combination, but we will score the original combination lower than the expanded, more predictive combination.

5. Save and name all useful feature value combinations

Feature value combinations can predict sponsorhood with extreme, high, moderate, low or no confidence – the prime case of the latter being a 50-50 split between sponsors and non-sponsors. All combinations that we find useful are saved and named in the interface shown in Figure 23.

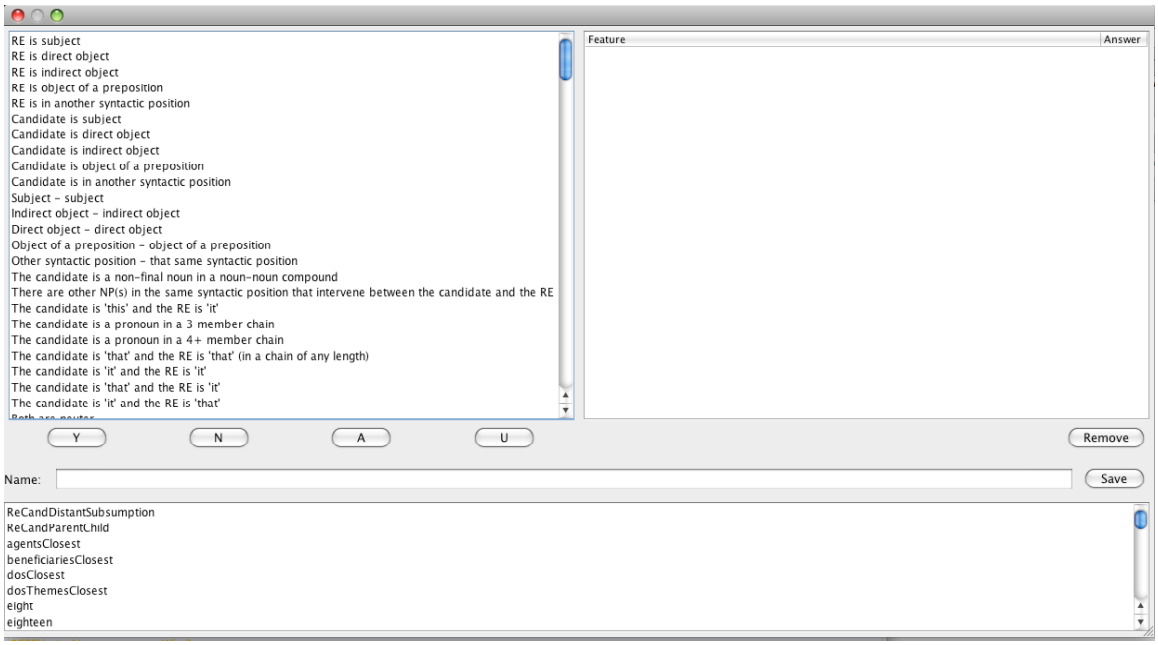

Figure 23. The interface for saving and naming feature value combinations.

6. Assign confidence levels to the feature value combinations.

We use a baseball metaphor for assigning confidence levels to feature value combinations: a "home run" has extremely strong predictive power, a "3rd base hit" has slightly less, a 2nd base hit still less, and a 1st base hit even less but still more than 50% based on our corpus. A screen shot of the categorization of features for our evaluation is shown in figure 24.

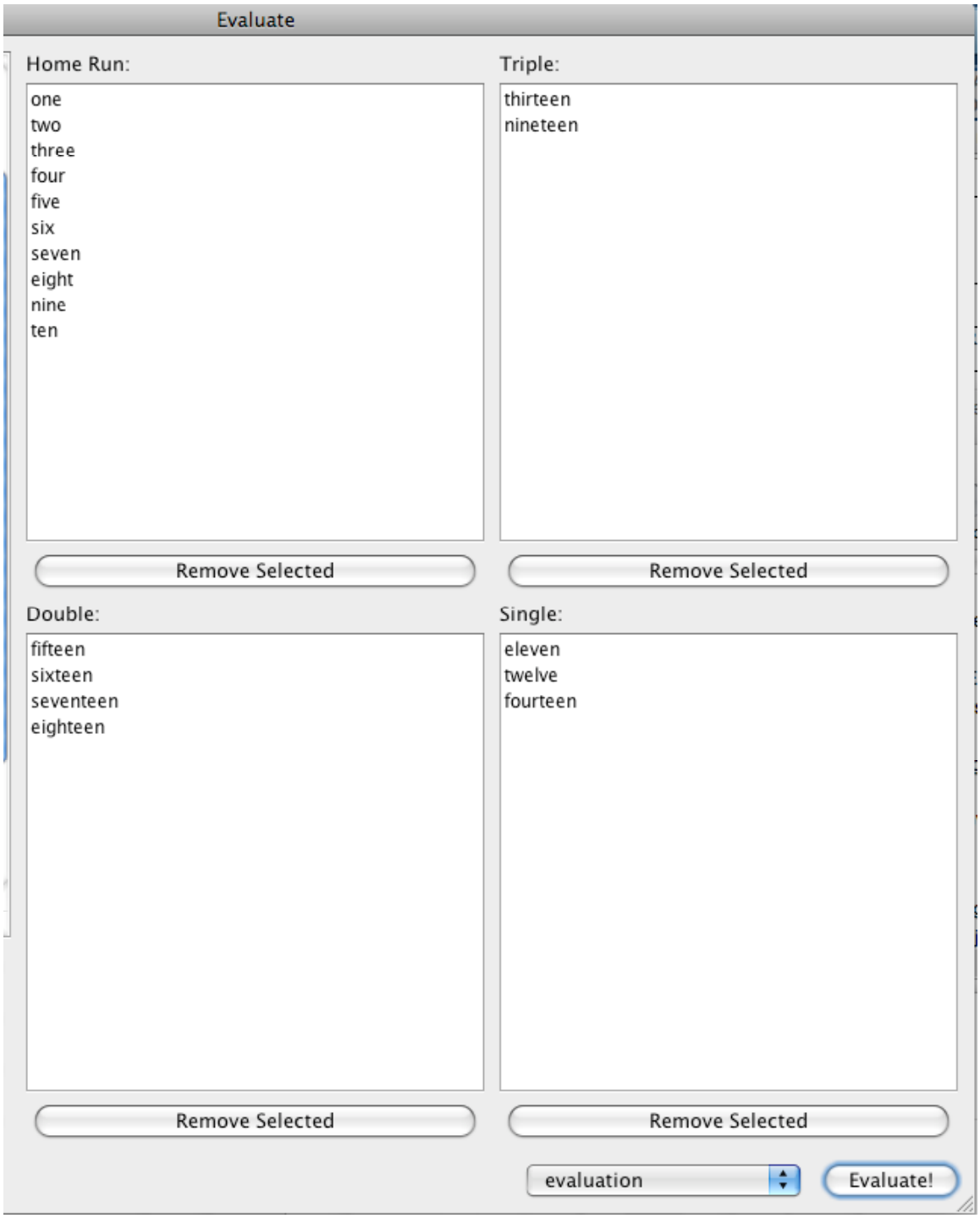

Figure 24. The scoring of the feature value combinations as used in our evaluation.

15 Interface Design, Data Storage

The annotation interface was designed and implemented in Java using the Swing library as a the GUI backbone. The interface connects to a secure port to complete a client-server style architecture, wherein all annotations are stored centrally in one database, and all requests of the database are passed through a custom Java API to cleanly handle the interaction with the data. In other words, the client interfaces are used as terminals; no actual processing or storage occurs on the client side.

The clients are designed to be very flexible in their display to the annotators. The nature of the project meant that we would want to rapidly, and often, change the available features to the annotators. We may want to simply rename a feature, or in a more complex operation, add, remove, or relocate large numbers of features. Having the interface hardcoded with these features (as we did in our original implementation, which was then superseded by a completely new one) would have been arduous and time-consuming to maintain, so we constructed a system that used the database to automatically render the latest version of the interface. Moreover this system allowed a superuser to completely change the available features without having to write a line a code.

Our design entailed constructing a database schema to store the desired features in a series of categories (for organizational purposes). This schema is described below. The interface itself, after loading and connecting to the server, requests an object-level view of the categories and features. Taking this object, it automatically constructs a Swing renderable interface from the contents of the database. This interface can then be used to support annotations that contain pointers to the feature elements of the interface schema. As a result, we were able to construct a secondary interface that a non-programmer could use to manipulate the main interface. Adding features, moving them between categories, and generally changing any of the features to be more in line with the current research can all be done through this interface, which causes immediate updates to the database, thus giving the annotators the latest without the need to involve programmers and or redistribute a new build.

Database Schema

The database system we used was PostgreSQL 7.4, which uses a fairly standardized SQL interpreter. Due to the nature of the interface, there are three main sections of the database: one for storing the features that the interface displays, a second for storing the actual results of annotation, and a third to facilitate investigation of the results. Naturally the three are interconnected, but they are conceptually distinct enough to separate for discussion. The SQL used to generate the tables is included in the index below.

Interface Schema

The interface is automatically constructed by querying the database for a series of categories, questions, and descriptions, the results of which are turned into Java Swing components, as described previously. The schema used to store and manipulate these objects in SQL consisted of two tables, *categories* and *questions*.

The purpose of the *categories* table is to provide structure to the interface, visually. By clustering questions into a folder, the user can more easily navigate the interface. The

elements of the *categories* table are recursively defined. Each can point to another category as its own parent, thus a tree structure can be constructed from a single table of category elements.

The *questions* table describes each question (or feature) in the interface: where the question lies (directly under a category, or under another question as a "follow-up"), a description of the question itself, examples if needed, a rule set for how and when to display follow-ups, and the ability to temporarily remove a question from the interface without losing its content. Importantly, each question has a unique id, which is referenced by the annotated data in the second schema.

Results Schema

The results are stored in a series of tables that define an annotation overall, which includes: the text; all references, each with a full set of answered questions (filled features); and other meta-data collected by hand. Again written in SQL, this schema consists of five tables, *annotations*, *annotation_references*, *reference_data*, *chain_elements*, and *eos_markers*.

The encompassing object in this schema is the *annotations* table. Each annotation contains the original text, a source and corpus, and a distinct id. A list of references (in the *annotation_references* table) points up to an individual annotation. These references contain the referring expressions, sponsors, and a host of other meta data. A list of answers to the various questions (values for the features) for each reference are found in the *reference_data* table, which contains a pointer to the unique id of the question, and the annotator's answer. This schema facilitates the design and redesign of the interface questions themselves, without the loss of data. Additionally, more meta data for each annotation is available in the *chain_elements* and *eos_markers* tables. Each contains a list of similarly structured objects that relate to a single annotation.

Investigation Schema

This schema is designed to support analysis of the results and deals primarily with setting up feature value combinations and sorting annotations into groups to be analyzed. The schema consists of three tables, *hypotheses (i.e., feature value combinations)*, *folders*, and *annotations_in_folders*. Each hypothesis is a named entity consisting of a list of questions (referenced by their unique id) and an expected answer. For the purposes of investigation, a hypothesis (or many) would be run over a collection of texts that are grouped into a folder for ease of use. This requires the use of two tables, one to store the folder's meta data, and a second to store a list of each annotation (by unique id) that was in a folder. This schema allows for an annotation to be in multiple folders at once without duplicating (and thus complicating) the data.

Database Schema SQL Index

```
CREATE TABLE categories
(
  id bigserial NOT NULL,
  category_name text,
```

```
  parent_id integer, -- -1 means root (no parent)
  hidden boolean,
  CONSTRAINT categories_id_key PRIMARY KEY (id)
)
WITHOUT OIDS;
ALTER TABLE categories OWNER TO ilit;
COMMENT ON COLUMN categories.parent_id IS '-1 means root (no parent)';

CREATE INDEX categories_category_name_index
  ON categories
  USING btree
  (category_name);

CREATE TABLE questions
(
  id bigserial NOT NULL,
  description text,
  category_id integer,
  question_id integer,
  open_on_yes boolean,
  open_on_no boolean,
  open_on_ambiguous boolean,
  open_on_unknown boolean,
  hidden boolean,
  example text,
  CONSTRAINT questions_id_key PRIMARY KEY (id)
)
WITHOUT OIDS;
ALTER TABLE questions OWNER TO ilit;

CREATE INDEX questions_category_id_index
  ON questions
  USING btree
  (category_id);

CREATE INDEX questions_description_index
  ON questions
  USING btree
  (description);

CREATE INDEX questions_question_id_index
  ON questions
  USING btree
  (question_id);

CREATE TABLE annotations
```

```
(
  id bigserial NOT NULL,
  original_text text,
  source text,
  corpus text,
  CONSTRAINT annotations_id_key PRIMARY KEY (id)
)
WITHOUT OIDS;
ALTER TABLE annotations OWNER TO ilit;

CREATE INDEX annotations_original_text_index
  ON annotations
  USING btree
  (original_text);

CREATE TABLE annotation_references
(
  id bigserial NOT NULL,
  annotation_id integer,
  referring_expression text,
  referring_expression_index integer,
  sponsor text,
  sponsor_index integer,
  is_valid_sponsor boolean,
  sponsor_concept text,
  sponsor_selecting_verb_concept text,
  refex_concept text,
  refex_selecting_verb_concept text,
  was_auto_generated boolean DEFAULT false,
  CONSTRAINT references_id_key PRIMARY KEY (id)
)
WITHOUT OIDS;
ALTER TABLE annotation_references OWNER TO ilit;

CREATE INDEX references_annotation_id_key
  ON annotation_references
  USING btree
  (annotation_id);

CREATE TABLE reference_data
(
  id bigserial NOT NULL,
  references_id integer,
  question_id integer,
  answer text,
  CONSTRAINT reference_data_id_key PRIMARY KEY (id)
```

```
)
WITHOUT OIDS;
ALTER TABLE reference_data OWNER TO ilit;

CREATE INDEX reference_data_references_id_index
  ON reference_data
  USING btree
  (references_id);

CREATE TABLE chain_elements
(
  id bigserial NOT NULL,
  references_id integer,
  chain_element_text text,
  chain_element_index integer,
  CONSTRAINT chain_elements_id_key PRIMARY KEY (id)
)
WITHOUT OIDS;
ALTER TABLE chain_elements OWNER TO ilit;

CREATE INDEX chain_elements_references_id_index
  ON chain_elements
  USING btree
  (references_id);

CREATE TABLE eos_markers
(
  id bigserial NOT NULL,
  annotation_id integer,
  eos_marker integer,
  CONSTRAINT eos_markers_key PRIMARY KEY (id)
)
WITHOUT OIDS;
ALTER TABLE eos_markers OWNER TO ilit;

CREATE TABLE hypotheses
(
  id bigserial NOT NULL,
  hypothesis text,
  question_id integer,
  answer text,
  CONSTRAINT hypotheses_key PRIMARY KEY (id)
)
WITHOUT OIDS;
ALTER TABLE hypotheses OWNER TO ilit;
```

```
CREATE TABLE folders
(
  id bigserial NOT NULL,
  folder text,
  CONSTRAINT folders_key PRIMARY KEY (id)
)
WITHOUT OIDS;
ALTER TABLE folders OWNER TO ilit;

CREATE TABLE annotations_in_folders
(
  folder_id integer,
  annotation_id integer,
  id bigserial NOT NULL,
  CONSTRAINT annotations_in_folders_key PRIMARY KEY (id)
)
WITHOUT OIDS;
ALTER TABLE annotations_in_folders OWNER TO ilit;
```

Design Drawbacks

One of the major problems that came to light as a result of our database schema design was a lack of fine-grained control over the meta data of individual sentences in a text. Our approach was to maintain all of the original text for an annotation as one field contained in the *annotations* table. However, after the fact we realized that we needed a way to identify individual sentences for the purposes of hypothesis testing, as well as associating meta data (such as speaker changes). We had to inject the *eos_markers* table and add index markers to identify sentences within the text, as well as mash meta data directly into the text itself. This was further complicated by parsing errors and other related problems. In hindsight, we should have broken down the sentences from the outset into another *sentences* table, where each sentence could have its own associated meta data.

Feature Generation

We were able to develop scripts to run over all of the annotated texts. These scripts performed two main tasks: *candidate generation* and *feature generation.* We did not need to alter the existing DB schema in order to accomplish these tasks.

The *candidate generation* task involved determining text span strings that preceded referring expressions from the text meta-data that specified clause and sentence boundaries. These boundaries and the location of the referring expressions were stored as string indices. In order to create the candidates that were necessary, we calculated the beginning and end span boundaries, and stored them as new candidates, with no feature data. The *feature generation* task marked all of the candidates, both the manually annotated, and automatically generated. Where in the previous task the string boundaries presented no real problem, in this task they dramatically complicated issues. Many of the generated features required that a candidate's string boundaries be compared to the

boundaries of the meta-data for sentence and phrase boundaries. Difficulty arose because there were a large percentage of cases where an annotator would have accidentally specified leading or trailing spaces, thus throwing off these comparisons. This leading white space was automatically trimmed by a script. In all, we learned that is best to avoid using string indices at all costs, even if it seems nothing more should be necessary at the start of a project.

Testing these scripts was done in two main steps. First, a test annotation was created that contained many features and candidates that should be created by the scripts. The scripts were run on this single text, and their outputs verified. The test text was modified to check other features, and this process repeated. Next, the tester ran the scripts on a large subset of the texts. During this run, the tester would inspect the outputs and verify output, correcting any errors. This process was complicated by annotator error, which would often be difficult to discover.

16 Evaluation

Before moving to the evaluation findings, a few words about the nature of evaluating this kind of data.

First, the challenge is not simply to find the correct NP sponsor among many candidate NPs or to find the correct text span sponsor among many candidate text spans. The challenge is to find the correct sponsor, not knowing from the outset whether that sponsor is an NP or a text span. (I'm talking in terms of syntax here because the annotation was at the syntactic level; of course, talking in terms of semantics makes more sense and I will shift to that in future writing.) This means that when groups of feature value combinations are scored, they must be scored keeping in mind that text span sponsors will be compared against NP sponsors. Considering that only about 1/5 of contexts in our training and evaluation corpora had text span sponsors, the weights of feature value combinations involving text spans will probably need to be reduced to leave open the more probable choice of an NP sponsor. We have not pursued this comparative weighting in any depth, as mentioned above, for reasons of time.

Second, our training corpus was very small and many of the feature value combinations that we thought might hold great predictive power simply did not appear. If we were putting together a reference resolution system we might want to include those combinations with high scores on the basis of linguistic intuitions; however, here we did not pursue them.

Third, a large part of the evaluation of contexts with text span sponsors involved learning about the nature of text span sponsors, e.g., that adverbs and modalities often have to be stripped from a proposition to leave the actual sponsor. This is the linguistic work that will lie at the center of our upcoming small Robust Intelligence proposal.

Fourth, due to time constraints, we did not devote much time to actually trying different combinations of parameter values and different scorings for them to optimize a reference resolution engine. However, we did carry out an evaluation that shows proof of concept for this general approach to reference resolution and, more importantly, shows how convenient and robust the environment we developed is for the further study, testing and evaluation of all – including difficult – reference phenomena.

Finally, annotator errors were found during testing and evaluation. As mentioned earlier, each annotation was done by just one person, with students responsible for certain

key parts of the annotations, such as selecting the sponsor. We did not record all errors found and did not recheck all the feature values in the corpus prior to testing and evaluation.

Top-level descriptions of the training and evaluation corpora

Number of contexts in training corpus: 305

Number of contexts with an NP sponsor: 240

Number of contexts with a text span sponsor: 65

Percentage of contexts with a text span sponsor: 21

Average number of text span candidates generated for each example: 4.3

Average number of NP candidates encoded for each example: 3.975 (it would have been 4 if every context had had at least three non-sponsors)

******

Number of contexts in evaluation corpus: 153

Number of contexts with an NP sponsor: 125

Number of contexts with a text span sponsor: 28

Percentage of contexts with a text span sponsor: 22

Average number of text span candidates generated for each example: 4.6

Average number of NP candidates encoded for each example: 3.92

To find useful feature value combinations and determine what their scores should be (home run, 3rd base, etc.), we ran various queries over the training corpus to see how many candidates with a given feature value combination were actually the sponsor. For example, if 8/10 candidates were the sponsor, this combination has quite good predictive power, whereas if 49/104 candidates were the sponsor, it has no predictive power at all.

The feature value combinations for NPs and text spans are separate since NPs and text spans have different relevant features. For the NPs, we always kept 2 features fixed: the gender and number of the candidate and the RE had to match (these are relatively simple to determine based on surfacy processing and in the large majority of cases, if there is no gender and number match there is no coreference). So combinations of 3 feature values are actually gender match + number match + one other feature value, and so on. We

looked at combinations of 3 feature values mostly as a baseline, not expecting very good predictive power, and looked at select combinations of 4 and 5 feature values with more confidence of getting more predictive power. The drawback was that, in a small corpus, many combinations were not attested.

In the results presented below, the blue text surrounded by / / is just for human orientation; the parameter values above the table hold for all rows of the table; and the parameter values in the table are added to those above the table to create the full feature value combination. "Sponsors" indicate the actual number of sponsors with these feature values in the training corpus, and "total hits" indicates the total number of candidates with these feature values.

The candidates that match a combination of feature values need not *only* have those feature values, they can have others as well; those others are simply not being considered in the pattern at hand.

Here, we present only a sampling of the evaluation results. The full inventory can be found in the project report (McShane 2009b).

Combinations of 3 Feature Values for NPs

/Syntactic function matching/
Gender matches
Number matches AND

	sponsors	total hits
Syntactic function matches	50	79
Subject – subject	45	60
Direct object – direct object	5	13
Indirect object – indirect object	0	0
Object of a preposition – object of a preposition	0	6
There are other NP(s) in the same syntactic position that intervene	7 (4 of these have intervening animates in the same syntactic position, so they really are the closest sponsor available, making the count 3/23, not 7/23)	23

/Semantic likeness of RE and cand/
Gender matches
Number matches AND

	sponsors	total hits
RE and cand are synonyms	6	7

Combinations of 4 Feature Values for NPs

/Closest and {OR syntactic function matches, case role matches, semantically similar referring concepts, semantically similar selecting verb concepts}/

Gender matches

Number matches

NP cand is closest of the NP candidates AND

	sponsors	total hits
Syntactic function matches	21	26
Subject – subject	18	20
Direct object – direct object	3	4
Indirect object – indirect object	0	0
There are other NP(s) in the same syntactic position that intervene	1	1
Case role matches	7	11
Agent – agent	3	3
Theme – theme	4	8
Experiencer – experiencer	0	0
Beneficiary – beneficiary	0	0
RE and cand are synonyms	4	4

/Synonymous RE and cand and {OR syntactic function matching, case role matching or semantically related selecting verbs, distance}/

Gender matches

Number matches

RE and cand are synonyms AND

	sponsors	total hits
Syntactic function matches	3	4
Subject – subject	3	4
Direct object – direct object	0	0
Indirect object – indirect object	0	0
There are other NP(s) in the same syntactic position that intervene	0	1
Case role matches	0	0
Agent – agent	0	0
Theme – theme	0	0
Experiencer – experiencer	0	0
Beneficiary – beneficiary	0	0
selecting verbs for RE and cand are synonymous	0	0
selecting verbs for RE and cand are in parent/child relationship	0	0
selecting verbs for RE and cand are in distant subsumptive relationship	0	0
NP cand is closest of the NP candidates	4	4
NP cand is 2nd closest of the NP	2	2

candidates		
NP cand is 3rd closest of the NP candidates	0	0
NP cand is farthest away of the NP candidates	0	1

Combinations of 5 Feature Values for NPs

We're cutting down on the combinatorics here based on what we expect to be useful and not useful. In the future we can add more combinations if they are attested and prove useful.

/Syntactic function and case role match with different distances/
Gender matches
Number matches
Syntactic function matches
Case role matches

	sponsors	total hits
NP cand is closest of the NP candidates	5	6
NP cand is 2nd closest of the NP candidates	3	5
NP cand is 3rd closest of the NP candidates	3	4
NP cand is farthest away of the NP candidates	1	3

Lexico-Syntactic Patterns for NP Candidates

We expected patterns to be highly predictive but our corpus was too small to have many hits on them. Some can be easily searched for over the Web in future work; others require more involved parsing so creating an inventory of them will require more effort.

Some notes on patterns:

1. For the following there were **no hits**; the follow-up property values are not shown because they are moot (e.g., for "this is where" the follow-up property value is "the candidate is the preceding temporal expression").

The RE is in the "(then)…" clause of a "when…(then)…" statement
The RE is in the "(then)…" clause of an "if…(then)…" statement
NP-candidate: {prop with RE}
NP-candidate – {prop with RE}

Full prop ending with NP-candidate: {prop with RE}
Full prop ending with NP-candidate – {prop with RE}
Full-prop-candidate ending with NP: {prop with RE}
Full-prop-candidate ending with NP – {prop with RE}
Full-prop-candidate not ending with NP: {prop with RE}
Full-prop-candidate not ending with NP – {prop with RE}
The RE is in the "(then)…" clause of a "when…(then)…" statement
The RE is in the "(then)…" clause of an "if…(then)…" statement
Prop ending with "…this: Candidate."
That (this) is where
That (this) is who
That (this) is when
That (this) is what
That (this) is (is the reason) why
That (this) is how

Constructions of the type "this is where" can be easily searched for on the Web, providing a large corpus in a short time. We did not specially focus on these patterns in this project.

2. Some features need to be reworked in order to be able to be used properly in queries.

The candidate is a pronoun in a 3 member chain.
In defining this feature we included the RE as a chain member such that the candidate was necessarily the sponsor if this feature was checked as 'yes'. We need to change this feature to "the candidate is it/this/that" and remark all texts accordingly. This can be done automatically using a stoplist of pronouns. When we start marking texts for animate REs as well, we will need a new feature for those pronouns. (This feature was originally introduced when we were describing only sponsors, not non-sponsors as well, and it was inadvertently not changed when we shifted to describing both sponsors and non-sponsors.)

The candidate is a pronoun in a 4+ member chain.
Same problem as with the 3 member chain. The feature needs to be "the candidate is a pronoun whose sponsor is a pronoun".

3. The following patterns were also hardly attested in our training corpus:

	sponsors	total hits
The candidate is 'this' and the RE is 'it'	1	1
The candidate is 'that' and the RE is 'that' (in a chain of any length)	0	0
The candidate is 'it' and the RE is 'it'	6	6

The candidate is ‘that’ and the RE is ‘it’	0	0
The candidate is ‘it’ and the RE is ‘that’	0	0

	sponsors	total hits
The candidate is ‘this’ and the RE is ‘it’	0	0
The candidate is ‘that’ and the RE is ‘that’ (in a chain of any length)	0	0
The candidate is ‘it’ and the RE is ‘it’	1	1
The candidate is ‘that’ and the RE is ‘it’	0	0
The candidate is ‘it’ and the RE is ‘that’	0	0

4. Some patterns are actually descriptive properties that aren’t to be used directly to predict a sponsor. For example, the following 2 features show where the semantics of the RE come from, which is important from a linguistic standpoint but not really necessary in creating feature value combinations, unless we want to give more confidence to semantics drawn from different sources (e.g., we trust the semantic information provided by a nominal in an existential construction than by an adjective in an existential construction).

- Existential construction whose predicate nominal shows the meaning of the RE
- Existential construction whose predicate adjective shows a useful constraint on the meaning of the RE

Text Span Candidates

Most of the work carried out on text span candidates was studying their properties, not creating and scoring feature value combinations that use them, since our corpus was not sufficient to be very representative for that. The project report show some initial results about text-span sponsors.

The Scored Feature Value Combinations Used for Evaluation

We only created, scored and used a subset of the possible feature value combinations in this evaluation. The point of the evaluation was to show proof of concept and to show that our environment is working end-to-end, including the automatic evaluation functions.

The feature value combinations we used and their scores – determined manually in consultation with the results of testing presented above – are as follows. We used the raw scores blindly in this evaluation to predict the baseball scores to be used in the reference resolution engine, but linguistic intuitions could also be used to modify scores and add more scored patterns to the inventory, even if they were not attested in the training corpus.

< 63% - pattern not used
63-74% - 1
75%-89% - 2
90-95% - 3
100% - 4

number as name	combo	raw score	baseball score
one	gender matches number matches agent-agent	4/4=100%	hr
two	gender matches number matches NP cand is closest of the NP candidates agent-agent	3/3=100%	hr
three	gender matches number matches NP cand is closest of the NP candidates RE and cand are synonyms	4/4=100%	hr
four	gender matches number matches NP cand is 2nd closest of the NP candidates RE and cand are synonyms	2/2=100%	hr
five	gender matches number matches Selecting verbs for RE and cand are synonymous subject-subject	2/2=100%	hr
six	Gender matches Number matches Selecting verbs for RE and cand are synonymous case-role matches	2/2=100%	hr
seven	Gender matches Number matches	4/4=100%	hr

	Syntactic function matches agent-agent		
eight	Gender matches Number matches Subject-subject agent-agent	4/4=100%	hr
nine	The candidate is ‘it’ and the RE is ‘it’	6/6=100%	hr
ten	The candidate is ‘this’ and the RE is ‘it’	1/1=100%	hr
eleven	gender matches number matches syntactic function matches	50/79 = 63%	1
twelve	gender matches number matches NP cand is closest of the NP candidates direct-object – direct-object	¾=75%	1
thirteen	Gender matches Number matches Case role matches subject-subject	9/10=90%	3
fourteen	Text span cand is S1 speaker break before S0 speaker break before S1	4/6=67%	1
fifteen	gender matches number matches subject-subject	45/60 = 75%	2
sixteen	gender matches number matches RE and cand are synonyms	6/7=86%	2
seventeen	gender matches number matches NP cand is closest of the NP candidates syntactic function matches	21/26=81%	2
eighteen	Gender matches Number matches Subject-subject theme-theme	4/5=80%	2
nineteen	gender matches number matches NP cand is closest of the NP candidates subject-subject	18/20=90%	3

The feature value combinations are input into the evaluation interface as shown in Figure 25, grouped by their confidence levels.

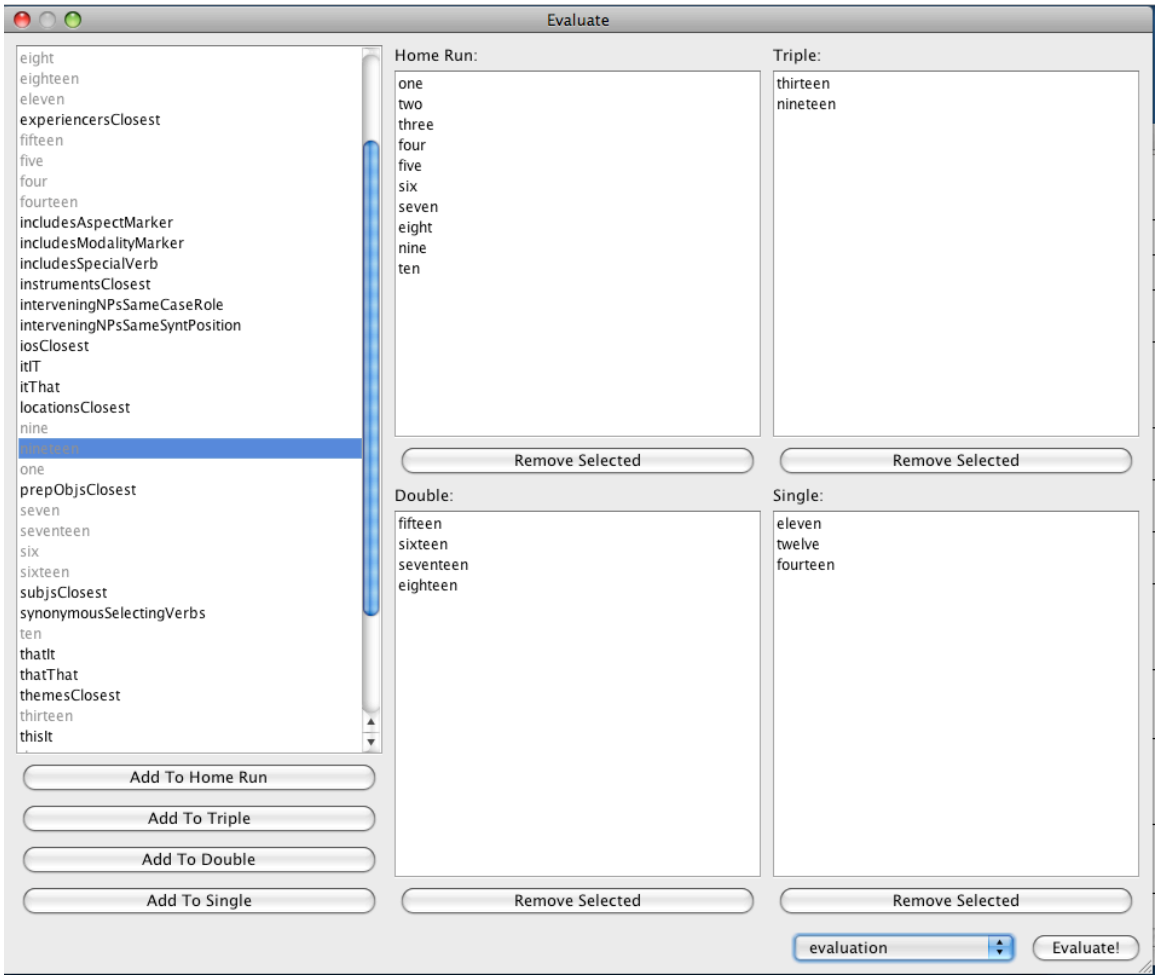

Figure 25. Selecting feature value combinations to be used in a reference resolver and assigning them scores based on their predictive power.

The results of automatic evaluation are shown in Figures 26 and 27. The evaluation statistics are at the bottom of the screen (Figure 26) and the hits associated with each feature value combination (called "hypotheses" on the screen) are shown to the right of the list of feature value combinations, when clicked on (Figure 27).

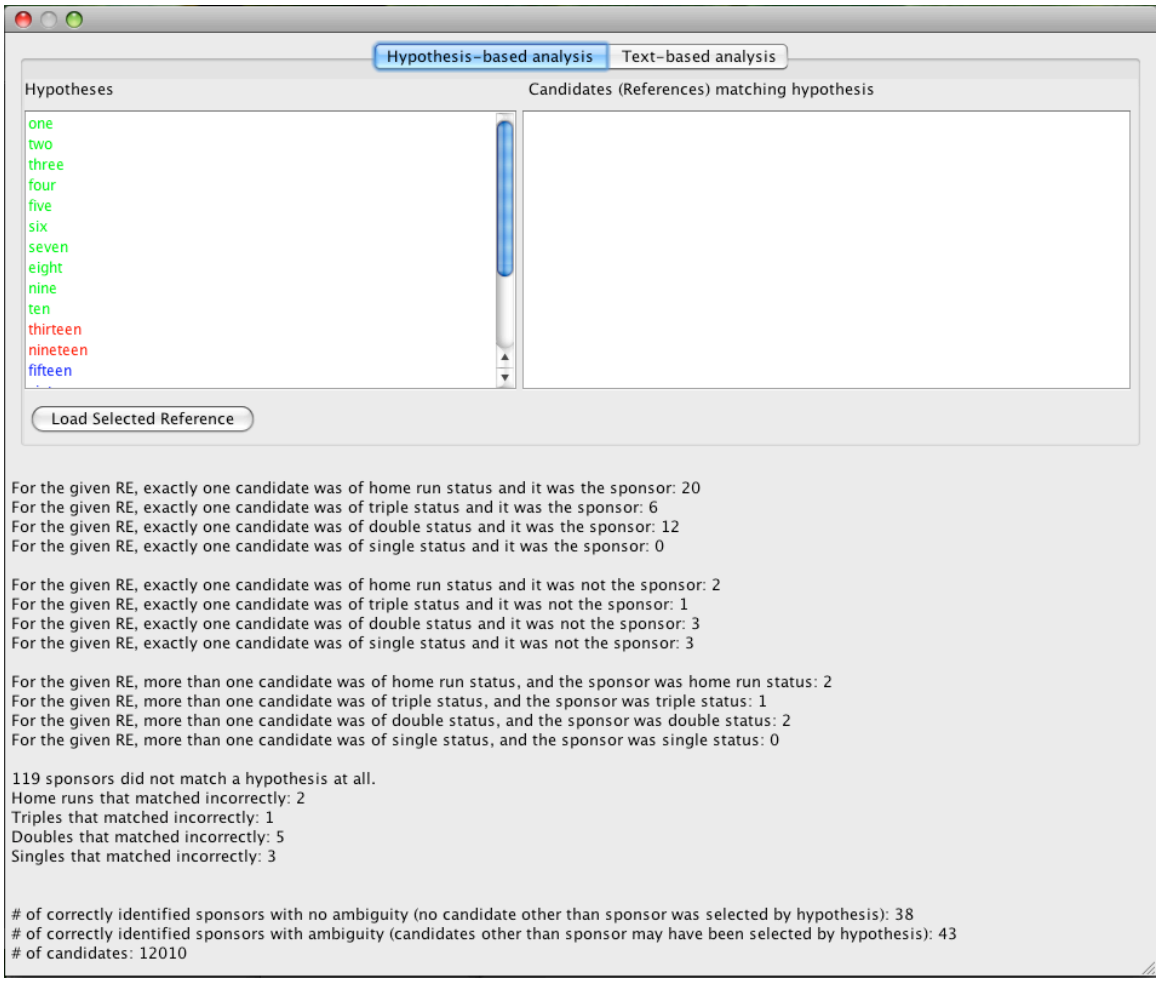

Figure 26. The results of evaluation.

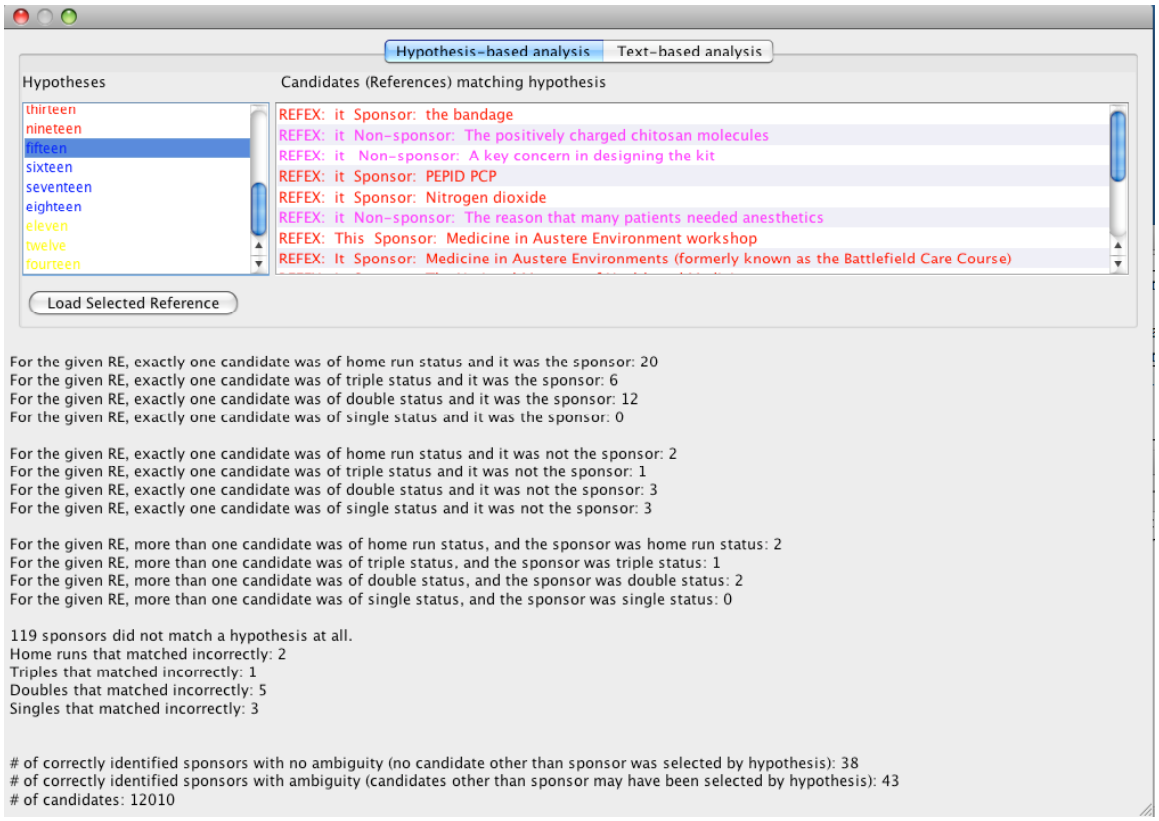

Figure 27. Looking at how many times feature value combination "fifteen" was used in the corpus. Red hits are actual sponsors; pink hits are non-sponsors.

A summary table of the use of each pattern is shown in the table below.

number as name	combo	baseball score	correct/hits in evaluation corpus
one	gender matches number matches agent-agent	hr	1/1
two	gender matches number matches NP cand is closest of the NP candidates agent-agent	hr	0
three	gender matches number matches NP cand is closest of the NP candidates RE and cand are synonyms	hr	2/2

four	gender matches number matches NP cand is 2nd closest of the NP candidates RE and cand are synonyms	hr	0
five	gender matches number matches Selecting verbs for RE and cand are synonymous subject-subject	hr	1/3
six	Gender matches Number matches Selecting verbs for RE and cand are synonymous case-role matches	hr	1/1
seven	Gender matches Number matches Syntactic function matches agent-agent	hr	1/1
eight	Gender matches Number matches Subject-subject agent-agent	hr	1/1 (same as for seven)
nine	The candidate is 'it' and the RE is 'it'	hr	18/18
ten	The candidate is 'this' and the RE is 'it'	hr	0
eleven	gender matches number matches syntactic function matches	1	0/1
twelve	gender matches number matches NP cand is closest of the NP candidates direct-object – direct-object	1	0
thirteen	Gender matches Number matches Case role matches subject-subject	3	2/3
fourteen	Text span cand is S1 speaker break before S0 speaker break before S1	1	0/2
fifteen	gender matches number matches subject-subject	2	10/14
sixteen	gender matches number matches	2	0

	RE and cand are synonyms		
seventeen	gender matches number matches NP cand is closest of the NP candidates syntactic function matches	2	4/5
eighteen	Gender matches Number matches Subject-subject theme-theme	2	0
nineteen	gender matches number matches NP cand is closest of the NP candidates subject-subject	3	6/6

The central results from this evaluation are:

- the tested inventory of feature value combinations resulted in 38/153 sponsors being uniquely and correctly selected, and another 5 being selected but not unambiguously (2 or more candidates receive the highest score among all candidate scores)
- 119 of 153 sponsors in the evaluation corpus did not match any combination of feature values. This was expected because few such combinations were included in this evaluation run.
- (as can be seen from the feature value combination testing interface, Figure 19, which provides additional data not currently output in the evaluation statistics) 125 of the 153 contexts in the evaluation corpus had an NP sponsor; 28 had a text span sponsor.
- the feature value combinations of home run status performed very well: only in two cases was a candidate with home-run features not the sponsor, but in those two cases it was because there was another candidate that also had home-run status.
- the feature value combinations of triple and double status also performed very well
- pattern nine, "The candidate is 'it' and the RE is 'it'" performed best, selecting 18 out of 18 sponsors.
- feature value combination number fourteen was the only one that targeted text span sponsors. It had two matches, both of which were incorrect. In both of those cases, the actual sponsor was an NP. This underscores the fact that, when resolving it/this/that, the challenge is not only to determine which is the best of the NP sponsors or the best of the text span sponsors, it is to determine whether the sponsor is an NP or a text span to begin with.
- of the 28 text span sponsors: in 10, the sponsor was S1 (the interface shows 8, but there were 2 parsing errors having to do with speakers and subtitles being understood as part of the following sentence per se); in 11 the sponsor was "part

of S1" (the interface shows 10 but one was missed for the same parsing reason described above); in 1 the sponsor was part of S0; in 1 the sponsor was part of S2 (S1 was an interjection which threw off the "contiguity" expectation); in 2 the sponsor was all of S2 and the contiguity expectation was defied for no easily explainable reason; in 1 the sponsors was S2&S1.

These results, and the environment we have developed that permit very convenient study of reference contexts, show proof of concept that this approach to learning about the behavior of referring expressions and configuring reference resolvers according to the strategy described here have promise and should be pursued further.

17 Summary of Accomplishments and Utility of the Work

1. We carried out **NLP-oriented study** of the referential properties **of pronominal *it/this/that*** with a focus on creating computer-tractable resolution heuristics. This involved studying a corpus of texts and reading associated descriptive and NLP literature.
2. We **created an inventory of features that is far broader** than those used in most systems. We say "broader" rather than "larger" because statistical systems can use very many features for a given feature type: e.g., the distance between the referring expression (RE) and the candidate sponsor could, in principle, be broken down into hundreds of features: one word away, two words, one phrase away, etc. The inventory of features that was compiled includes surface features (pre-processing features, morphological features, etc.), syntactic features, semantic features and discourse features. Only those features that can be automatically computed – even if imperfectly – were included. We used as a baseline for computability the OntoSem text processing system, since it manipulates practically all of the features used in other systems as well as quite a number of others. The point to emphasize is that we did NOT include features like "the candidate is the topic of the discourse" since there are currently no automatic methods (at least there are none described in the literature) to detect discourse topic with any reasonable degree of confidence.
3. We created **a new approach to automatically resolving reference that does not require a large annotated corpus for training.** Although this approach was developed for the difficult referring expressions (REs) *it/this/that*, for which no large annotated corpus exists, it is equally applicable to other kinds of REs. Very briefly, this approach involves combining groups of feature value combinations that are assigned confidence levels. For example, if a candidate has the features "matches the referring expression's syntactic position, matches the referring expression's semantic role, is the closest candidate", then there is a high likelihood that the candidate is the sponsor. This illustrative example intentionally includes features that are typical of reference resolution engines; however, many of the features used in the system involve deeper knowledge than is used in most other reference resolution systems. The feature values of each candidate sponsor (essentially, "antecedent", though we don't use that term) are matched against the

scored groups of feature value combinations, and the candidate that matches the highest scoring group is selected as the sponsor. If more than one candidate matches a highest scoring group, then more than one candidate is returned as the sponsor with the residual ambiguity not resolved.

4. We developed an inventory of **features that can be automatically derived from the elicited features** as well as a methodology for computing them.
5. We developed **a methodology for annotating texts** for our broad range of features. This strategy uses 3 "levels" of annotator: lightly trained undergraduates, more highly trained students (undergraduate or graduate), and superusers (linguists). This division of labor between annotators made the annotation as cheap as possible without asking students to carry out tasks that were beyond their level of training.
6. We developed an **annotation and evaluation environment** that includes **extensive functionalities** including but not limited to the following. Users can:

    (a) assign features to text elements by highlighting strings (e.g., making a string blue in the input text means it is the referring expression in question)
    (b) answer multiple-choice questions
    (c) fill in blanks with strings
    (d) create feature-value groups that are thought to have predictive power for resolving reference (e.g., "the candidate matches the referring expression's syntactic position, matches the referring expression's semantic role, is the closest candidate")
    (e) test those feature value groups on the training portion of the annotated corpus to see how often each appears on sponsors and how often on non-sponsors (i.e., how predictive of a sponsor the group is)
    (f) manually assign confidence scores to feature value groups based on the automatically derived results of how predictive they are of a sponsor; e.g., if the feature value group in (e) were to be present on a sponsor 29/32 times, then that group might be assigned the confidence level "3rd base" (we use a baseball metaphor as an unambiguous indication of how strongly predictive a feature value group is, with the confidence decreasing from home run to 3rd base to 2nd base to 1st base. By the time the feature group is at 1st base it is only slightly better than flipping a coin.)
    (g) analyze the contexts in which feature value groups that typically predict a sponsor failed to do so, and attempt to modify the group so that it will have more predictive power; this does not mean that we will get rid of the original group, but if we find a more predictive group – even if with less coverage – we might add it as yet another group with a higher score
    (h) group parts of the data base into training and evaluation sets
    (i) automatically evaluate how an inventory of hypotheses would process a corpus using many different metrics.

7. We **created an annotated corpus** divided into training and evaluation portions.

8. We developed an **inventory of feature value combinations with confidence levels** that have proven useful for resolving the referring expressions *it/this/that*.
9. We **evaluated our scored feature value combinations** on an evaluation corpus.

The results of this project have both short-term and long-term utility. For example:

- The approach and the environment can be applied to all kinds of referring expressions and any language with no modifications of any kind required, barring issues associated with non-Latin character sets (recall that users can change the inventory of parameters and values at will).
- The environment and the corpus we have already annotated provide significant data for the further study of the reference patterns of *it/this/that.*
- The environment is completely ready for further study and annotation of these and other referring expressions.
- The annotated corpus and our best-yet configuration of a reference resolver, as reported below, are available to other developers.
- The research carried out both has significant reportable content and has made clear the next steps, to be detailed in the small Robust Intelligence proposal to be submitted this winter.
- This work provides proof of concept that progress *can* be made in the near term on difficult referring expressions without the need to fulfil unrealistic preconditions, like annotating a large corpus for all kinds of referring expressions, from personal pronouns to demonstrative pronouns to definite descriptions (NPs with the determiner *the*) to bridging constructions.
- This work suggests that using a larger inventory of both knowledge-based and surface features can improve reference resolution.
- The sets of features and ways of determining or annotating their values that have been researched in this project will prove invaluable as inputs to machine learning studies concerned with finding the best combination of features for resolution of complex reference cases.
- Our analysis of the problems associated with annotating surface strings, rather than semantic structures, provides evidence that a semantic approach will ultimately be necessary for many of the more difficult cases, whether or not one can argue for a more surface-oriented approach to the simpler cases.
- We have annotated a corpus for the referring expressions it/this/that using a large inventory of features. The annotation effort was carried out in stages by annotators with different skill levels, so many more contexts were input into the system and partially annotated than were fully annotated. The training and evaluation corpus were composed of 305 and 153 contexts, respectively. The currently incomplete annotations must be completed by a superuser, who provides data about more linguistically sophisticated features.

References Cited

Beale, Stephen, Sergei Nirenburg and Marjorie McShane. 2003. Just-in-time grammar. Proceedings 2003 International Multiconference in Computer Science and Computer Engineering, Las Vegas, Nevada.

Byron, Donna K. 2004. Resolving Pronominal Reference to Abstract Entities**,** University of Rochester Computer Science Department Technical Report #815, January 2004.

Carbonell, Jaime G. and Ralf D. Brown. 1988. Anaphora resolution: a multi-strategy approach. COLING 1988: 96-101.

Lobeck, Anne. 1995. Ellipsis: Functional heads, licensing, and identification. New York, NY: Oxford University Press.

McShane, Marjorie. 2005. A Theory of Ellipsis. Oxford University Press.

McShane, Marjorie. 2009a. Reference resolution challenges for an intelligent agent: The need for knowledge. IEEE Intelligent Systems, vol. 24, no. 4, pp. 47-58, July/Aug. 2009.

McShane, Marjorie. 2009b. Advances in difficult aspects of reference resolution: Project Report for SGER. ILIT Working paper #03-09. Nov. 19, 2009.

McShane, Marjorie. 2009c. Subject ellipsis in Russian and Polish. In Special Issue on Partial Pro-Drop, Studia Linguistica, edited by Anders Holmberg, Cambridge University Press, 63(1): 98-132.

McShane, Marjorie and Sergei Nirenburg. 2003a. Blasting open a choice space: learning inflectional morphology for NLP. Computational Intelligence 19(2): 111-135.

McShane, Marjorie and Sergei Nirenburg. 2003b. Parameterizing and eliciting text elements across languages. Machine Translation 18(2): 129-165.

McShane, Marjorie, Stephen Beale and Sergei Nirenburg. 2004a. Some meaning procedures of Ontological Semantics. Proceedings of LREC-2004.

McShane, Marjorie, Stephen Beale and Sergei Nirenburg. 2004b. OntoSem methods for processing semantic ellipsis. Proceedings of HLT/NAACL 2004 Workshop on Computational Lexical Semantics, Boston, Mass.

McShane, Marjorie, Sergei Nirenburg and Stephen Beale. 2005a. Semantics-based resolution of fragments and underspecified structures. Traitement Automatique des Langues 46(1): 163-184.

McShane, Marjorie, Sergei Nirenburg and Stephen Beale. 2005b. An NLP lexicon as a largely language independent resource. Machine Translation 19(2): 139-173.

McShane, Marjorie, Sergei Nirenburg and Stephen Beale. 2008a. Resolving Paraphrases to Support Modeling Language Perception in an Intelligent Agent. In Proceedings of the Symposium on Semantics in Systems for Text Processing (STEP 2008), Venice, Italy.

McShane, Marjorie, Sergei Nirenburg and Stephen Beale. 2008b. Two Kinds of Paraphrase in Modeling Embodied Cognitive Agents. In Proceedings of the Workshop on Biologically Inspired Cognitive Architectures, AAAI 2008 Fall Symposium, Washington, D.C., Nov. 7-9.

McShane, Marjorie and Sergei Nirenburg. 2009a. Dialog Modeling Within Intelligent Agent Modeling. Proceedings of the IJCAI-09 Workshop on Knowledge and Reasoning in Practical Dialog Systems, Pasadena, Cal., July 12, 2009, pp. 52-59.

McShane, Marjorie, Sergei Nirenburg, Stephen Beale. 2009b. Ontological Semantic analysis and difficult reference resolution informing each other. ILIT Working paper #02-09. Nov. 19, 2009.

Nirenburg, Sergei and Victor Raskin. 2004. Ontological Semantics. MIT Press.