

# Controlling Linguistic Coreference in Graphical Interfaces

Jill Nickerson

Division of Engineering and Applied Sciences

Harvard University

Cambridge, MA 02138, USA

nickerso@eecs.harvard.edu

## Abstract

We propose innovational mechanisms for controlling linguistic coreference in graphical interfaces. The mechanisms extend a system that allows a domain expert to perform language-neutral knowledge editing by interacting with natural-language text and produces texts in multiple languages based on the resulting knowledge base. The mechanisms address four challenges: 1) identifying algorithms for locating potential coreference relations, 2) using discourse structure to rank coreference candidates, 3) developing mechanisms for reaching a consensus between the expert and system with regard to coreference relations, and 4) examining different ways to represent these relations in the interface. We use novel methods to locate and represent potential coreference relations and to rank coreference candidates. We plan user studies to evaluate our methods.

## 1 Introduction

Representing the content of a document in a formal knowledge base (KB) enables documents expressing that content to be generated in multiple languages using the same underlying representation. Usually, building a KB requires an expert in the knowledge to be contained in the KB *and* an expert in the knowledge representation (KR) language being used to build it. The WYSIWYM (What You Say Is What You Meant) knowledge-editing method (Power et al., 1998), however, allows a domain expert to build and extend a KB representation without previous exposure to computational linguistics or KR languages. In WYSIWYM, the expert interacts with natural language, rather than a KR language, to build a KB that is used for multiple language generation of documents. A number

of practical computational linguistic problems must be solved for WYSIWYM to function effectively. Coreference is one of them.

Building a KB often requires a user to refer to entities that already have a representation in the KB. (In this paper, we use the term “entity” for a real world object, “KB object” for a representation in the KB that refers to an entity, and “linguistic expression” for a phrase generated by the system to refer to an entity.) If the system does not recognize references to the same entity, the KB will be inaccurate, e.g., it might contain multiple KB objects for the same entity. Since a correct KB is necessary to generate texts with correct referential forms, it is essential that the system and user agree on references to a single entity. We describe innovational mechanisms that allow the system to communicate with a domain expert, who is assumed not to be an expert in computational linguistics, to resolve potential coreference relations.<sup>1</sup> After these relations are resolved, the mechanisms accurately represent them in the user interface (UI).

Throughout this paper, we use examples from our test domain of vehicle service manuals. Figure 1 shows a partially specified manual entry. The user encoded the knowledge that ‘drain hoses’ must be installed [C1]. He may wish to refer to the same ‘drain hoses’ to assert that they must be taped into position [C2]. If the system understands that the two mentions of the drain hoses corefer, it will generate coherent text with correct referential forms.

---

<sup>1</sup> Our coreference resolution task differs from the one often described in the literature (e.g. (Baldwin, 1997)). Coreference resolution in the traditional sense requires that a system, using only a text, determine which expressions have the same referent. In our system, the user and system must also agree on which expressions have the same referent, but some of the burden shifts to the user; the system, having identified a potential corefering expression, proposes several entities as coreference candidates and then solicits feedback from the user to determine if he wishes to refer to one of these entities again.

[A] To replace *some part* for the 2001 Buick Le Sabre:  
First perform the removal method:

[B] 1. Remove *some part* with a flat bladed tool by  
*some method*.

2. Do *some action* by *some method*.

Then, perform the installation method:

[C] 1. Install the drain hoses with *some tool* by  
*some method*.

2. Tape *some part* into *some position* by  
*some method*.

3. Fasten the transmitter battery into  
*some position* by *some method*.

Figure 1. Partially Specified Manual.

Previous work involving WYSIWYM (Van Deemter and Power, 1998) has investigated coreference in a small-scale application. Unlike previous work, we use new method to locate potential coreference relations and candidates, discourse structure to rank coreference candidates, and a wider variety of linguistic elements to represent coreference relations. Our methods scale better than previous approaches.

After briefly explaining WYSIWYM, we discuss our approach to extending it to handle coreference, including methods for locating potential coreference relations and using discourse structure to rank entities. We also examine issues related to the UI and present an example coreference mechanism.

## 2 WYSIWYM

WYSIWYM (Power et al., 1998) allows a domain expert to build a KB directly through interacting with natural-language “feedback text.” The feedback text, which is generated using a partial KB, communicates the current state of the KB and ways to extend it. Because the expert is directly editing a KR, the system does not need to interpret text. In addition, it is possible to use the same KB to generate texts in multiple languages, alleviating the need to translate a document from one language into others.

WYSIWYM has been deployed in several systems including DRAFTER for generating software documentation (Power et al., 1998) and PILLS for generating patient leaflets (Bouayad-Agha et al., 2002). We used WYSIWYM and domain knowledge gleaned from a corpus study of vehicle manuals (GM, 2002) to develop AUTO, a system technical writers can use to automatically generate vehicle manuals. The AUTO KB is like those of other systems using WYSIWYM.

In the feedback text for a partially specified manual generated by AUTO in Figure 1, bold, italicized words indicate locations where knowledge may be added. When the expert clicks on one of these phrases, he is presented with a menu of choices for extending the KB. For example, clicking the phrase “some action” [B2] results in a menu of valid actions such as remove, install, tape, and fasten. After the expert selects an action, the feedback text is updated to reflect the revised KR. When the expert has completed adding knowledge, the system can construct the “output text,” the actual manual text that does not include markings to indicate where knowledge may be added.

## 3 Coreference Mechanisms for Knowledge Editing

AUTO was designed to allow WYSIWYM<sup>2</sup> to handle coreference. In this paper, we focus on coreferring noun phrases in the feedback text. AUTO 1) identifies potential coreference relations, 2) uses discourse structure to rank the entities most likely to participate in these relations, 3) interacts with the user to reach an agreement on coreference relations, and 4) clearly represents these relations in the UI.

We are experimenting with coreference mechanisms that vary according to the following properties: type interpretation for locating potential coreference relations and identifying coreference candidates (Section 3.1), method used to rank these candidates (Section 3.2), and communication style and content of the UI (Section 3.3).

### 3.1 Type Interpretation

We provide for type to be defined in two ways: broad and narrow. The broad definition uses general types, i.e. the top of the hierarchy, such as “some part.” The narrow definition uses specific values of broad types, i.e. leaves of the hierarchy, such as “drain hoses,” a type of part. Previous work (Van Deemter and Power, 1998) has considered only narrow type. We experiment with systems using either broad or narrow type to identify coreference relations and candidates for coreference.

---

<sup>2</sup> This paper is mainly concerned with the inserting editing operation in WYSIWYM. In addition to inserting, van Deemter and Power (1998) also address copying and pasting.

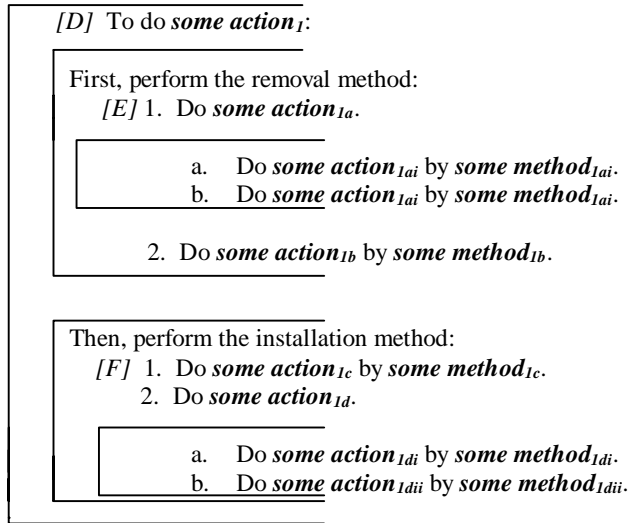


Figure 2. Replacement Procedure with Discourse Segments Delineated.

When *broad* type is used, all locations in the text which require an argument of the same broad type as an argument whose narrow type has already been specified represent locations where coreference is possible. For example, in Figure 1, locations where an argument of type ‘part’ has yet to be specified ([A], [B1], and [C2]) are locations at which it is possible to refer again to the drain hoses or the transmitter battery, ‘part’ arguments that have already been specified ([C1]; [C3]).

When *narrow* type is used, the system waits until a narrow type is specified before identifying potential coreference sites. For example, in Figure 1, the system waits until the user specifies ‘drain hoses’ as the value of ‘some part’ in [A], [B1], or [C2] before identifying that the value of one of these arguments may corefer to the drain hoses in [C1]. The decision of which interpretation to use is an empirical one that we plan to address with user studies.

### 3.2 Ranking Coreference Candidates

We rank the entities that may take part in a coreference relation using the discourse structure of the current manual before presenting linguistic expressions referring to them in menus. However, because any entity may be referred to again, the menus contain all entities of the appropriate type that have already been specified. Unlike speech and English text, in which information is processed from left to right, in knowledge editing, entities can be introduced in any order. Therefore, menus for knowledge that must be specified early in a

document contain linguistic expressions for entities that have already been specified later in the document.

Figure 2 shows the hierarchical discourse structure of a partially specified manual. The foundation for representing discourse structure as a hierarchy of segments comes from Grosz and Sidner (1986). For each potential coreference relation that has been identified, discourse structure is used to rank coreference candidates. Entities of the same type appearing in the same segment are deemed most likely to be referred to again. For example, if an entity of type  $\theta$  has been specified in both [E1a] and [E2], the system predicts that if [E1b] contains another reference to an argument of type  $\theta$ , the user will most likely refer to the entity mentioned in the current segment, [E1a], even though, in terms of linear recency,<sup>3</sup> both [E1a] and [E2] are equidistant from [E1b]. When entities of the same type in the current segment have been exhausted, the algorithm looks to daughter segments and then sibling segments before ascending the levels of the hierarchy in search of coreference candidates. Remaining objects are ranked according to linear proximity. We hypothesize that as the number of KB objects of the same type increases, users will prefer coreference mechanisms that rank coreference candidates. Because we expect that users are more likely to refer repeatedly to entities contained in instruction steps about the same topic, we expect that they will find menus that contain these entities first more manageable.

### 3.3 Communication Style and Content of the UI

#### 3.3.1 Communicating with the User to Determine Coreference Relations

The UI must facilitate the communication of the user and system coming to agree on coreference relations. We investigate two interaction styles, *direct manipulation* and *dialogue oriented*.

With direct manipulation interaction, each linguistic expression that refers to an entity has a “faithful copy” operation associated with it. If the user copies and then pastes the expression in a location where an entity of the same broad

<sup>3</sup> We measure linear proximity according to the number of noun phrases between the argument currently being specified and the potential coreference candidate.

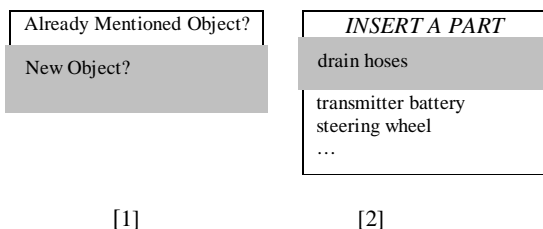


Figure 3. Menus for Specifying “*some part.*” Gray Indicates a Selected Item.

type must be specified, the system assumes that the user wishes to refer to the entity again. The system adheres to the user’s commands, extending the KB and generating text to represent the coreference relations that have been specified.

With dialogue interaction, the system presents the user with menus from which selections are made. Those entities hierarchically recent in the discourse structure to the one being specified are presented first in menus. The dialogue interaction style can deploy two methods for presenting coreference candidates: *status first* and *status concurrently*. When *status first* is used, the user specifies whether the object is new or already mentioned before examining coreference candidates. *Status concurrently*, however, displays menus containing both new and already mentioned objects together. Each method can be used with either broad or narrow type.

When **status first—broad type** is used, the user is first asked if the object is new or already mentioned (Figure 3 [1]). If the user chooses ‘new object,’ he then chooses from a list of all possible narrow types. Figure 3 [2] shows a sample menu containing narrow types of parts. If he chooses ‘already mentioned object,’ next, he chooses from a list of existing narrow types; and then, he chooses from a list of instances of the chosen narrow type. When **status first—narrow type** is used, the user first chooses from a list of all possible narrow types (Figure 3 [2]). Then, he is asked if the object is new or already mentioned (Figure 3 [1]). If he chooses ‘already mentioned object,’ he then chooses from a list of instances.

When **status concurrently—broad type** is used, the user chooses from a menu of all new and already mentioned objects of the broad type being specified. When **status concurrently—narrow type** is used, the user first selects a narrow type (Figure 3 [2]). Then, he chooses from a menu of new and already mentioned instances.

Direct manipulation and *status first* dialogue interaction with narrow type were first proposed

- [G] 1. Install the bolts with *some tool*.  
 2. Tighten them to *some position*.

Figure 4. Partial Instruction Set with Pronouns.

by van Deemter and Power (1998), while *status concurrently* is a new method. We plan user studies to determine the tradeoffs involved in direct manipulation and the two dialogue-oriented methods. Some experimental issues we plan to address include: efficiency and coreference errors (e.g., creating a new KB object when an existing one should have been reused and reusing a KB object when a new one should have been created). We expect that users will prefer *status concurrently* with narrow type for specifying coreference. Like *status first—narrow type*, this method forces the user to be as specific as possible before he is presented with coreference candidates, but unlike *status first—narrow type*, it allows him to reach his desired selection more quickly. This has been shown to be a desirable property for a UI (Shneiderman, 1992; Johnson, 2000). Though *status concurrently—broad type* allows the user to reach his destination more quickly, we expect that as more KB objects of the same type are introduced, menus will become too long when this method is used.

### 3.3.2 Representing Coreference Relations

The system must effectively distinguish new entities from existing ones. Representing coreference is important in the menus presented to the user and the feedback text representing the KB. The menus and feedback text may use different elements for representing coreference. We allow two types of elements to be used for representing coreference relations in the UI: *linguistic* and *artificial*.

The linguistic elements are pronouns determiners. Pronouns are used to refer to already mentioned objects. We use pronouns only in very constrained environments to avoid ambiguity. For example, if the communication is in English, we use pronouns only in the following situation: (1) a previous mention of the entity is in the preceding sentence, and (2) the previous and repeated mention of the entity are the first argument of an action.<sup>4</sup> In Figure 4, ‘them’ is used to refer to ‘bolts’ [G2] because a previous

<sup>4</sup> We may also investigate the use of centering theory for generating referring expressions (Grosz et al., 1995; Kibble and Power, 2000).

Type Interpretation [Section 3.1]	Broad
Method Used to Rank Coreference Candidates [3.2]	Hierarchical Discourse Structure
Communication Style [3.3.1]	Dialogue Oriented
Method Used to Present Coreference Candidates [3.3.1]	Status First
Coreference Relation Representation in Menus [3.3.2]	Determiners and Indices with Brackets
Coreference Relation Representation in Feedback Text [3.3.2]	Determiners and Indices with Brackets

Table 1. Parameter Values for Coreference Mechanism in Figure 5.

mention of bolts is found in the preceding sentence [G1], and ‘bolts’ is the first argument of install [G1] and tighten [G2].

When determiners are used, the system uses the definite article “the” to refer to *existing instances* of narrow types. If more than one instance exists, ordinal numbers are used. For example, if a manual contains two instances of type “tie wrap,” the system uses “the tie wrap” to refer to the first one (in terms of surface order) and “the second tie wrap” to refer to the second one in the feedback text and menus. When an instance of a narrow type already exists, the system uses the difference words “other” or “another” to refer to *new instances*, unspecified instances have no KB objects associated with them. For instance, the system uses the expression “another tie wrap” for a menu selection representing a new tie wrap instance. When an instance of a narrow type does not currently exist, no determiner is used.

The artificial elements that we use to represent coreference relations are indices with brackets. When indices with brackets are used, noun phrases in the text with the same index corefer. For example, all occurrences of [bolts]<sub>1</sub> in the text refer to the same entity.

Though previous work has used definite descriptions with ordinals in the output text (Power, 1999), coreference relations in the feedback text have only been represented using artificial elements (Van Deemter and Power, 1998). We expect that because linguistic elements lead to more coherent and natural text, users will prefer menus and feedback text containing linguistic elements as opposed to menus and feedback text containing only artificial elements. However, linguistic elements such as pronouns can introduce ambiguity and may not be as precise as artificial elements. Above all else, the feedback text must be clear to ensure that the correct output text is generated; therefore, we expect that, overall, users will prefer menus and

[H] To replace *some part* for the 2001 Buick Le Sabre: First perform the removal method:

- [I] 1. Disconnect [the tie wrap]<sub>1</sub> with *some tool*.
- Cut the transmitter from *some part* with *some tool* by *some method*.
  - Remove the tape from *some part* by *some method*.
2. Remove *some part* with *some tool* by *some method*.

Then, perform the installation method:

- [J] 1. Tape [the second tie wrap]<sub>2</sub> into *some position* by *some method*.
2. Follow *some steps*.

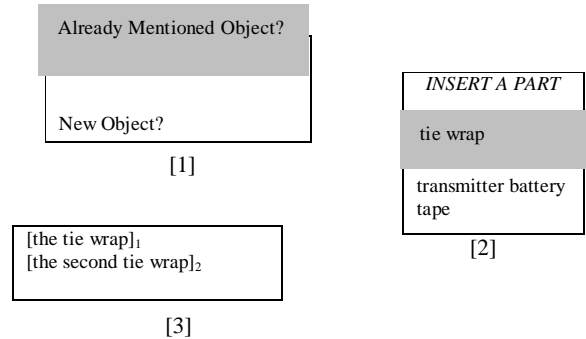


Figure 5. Partial Instructions Generated by AUTO and Menus for Specifying “*some part*” [I2].

feedback text with both linguistic and artificial for representing coreference. When both types of elements are employed, users use artificial elements to determine coreference relations in situations where the linguistic elements are ambiguous.

## 4 Example Coreference Mechanism

Figure 5 shows an interaction with a coreference mechanism using Table 1 parameter values. The user is in the process of specifying the value of ‘some part’ in [I2]. Clicking on the phrase ‘some part’ results in menu [1] in Figure 5. After the user chooses ‘already mentioned object,’ narrow types representing valid repeated mentions, ranked using discourse structure, are presented [2]. The user chooses to insert a part of type “tie wrap.” The final menu [3] presents the user with instances of this type, again ranked according to discourse structure, that he can refer to again. Determiners and indices with brackets are used in the menus and feedback text to represent coreference relations.

## 5 Ongoing Work

We have built the ontology for the vehicle manual domain. Generators for authoring manuals have been constructed in English and Spanish. We are implementing and evaluating our coreference mechanisms. We plan to extend AUTO to create a library of procedures that may be inserted within a new set of instructions without respecification. This capability would present challenges for our coreference mechanisms. If an entity of the same type is referred to in both an already generated procedure and the procedure currently being generated, the system must know if these entities are the same to extend the KB.

We plan to extend our coreference mechanisms to handle association/bridging reference (Clark, 1977) or implicit focus (Grosz, 1977) in which the referent is not mentioned explicitly.

Finally, we plan to investigate other methods for ranking menu options that combine discourse structural knowledge and domain knowledge. For example, if the action “tape” has been specified, and the system, after consulting a corpus or library of already generated procedures, determines that steering wheels are a type of part that are not commonly taped, steering wheel would be moved to the bottom of the menu, and entities that are likely to be taped would appear closer to the top of the menu. In addition to ranking coreference candidates using discourse structure, we plan to rank them according to the order in which KB objects are created and other theories of accessibility (e.g. (Cristea et al., 1998; Walker, 2000)).

## 6 Conclusion

We have described mechanisms for linguistic coreference in graphical interfaces. The mechanisms generate texts with correct referential forms by locating potential coreference relations and the entities that may take part in them, presenting these coreference candidates to the user, and then clearly representing coreference relations in the UI. We introduced new methods for locating and representing potential coreference relations and ranking coreference candidates. We plan user studies to evaluate our methods and to characterize the situations in which different methods are preferred.

## Acknowledgments

This work is supported in part by NSF grant IIS-9811129. We thank the people at the Univ. of Brighton for providing WYSIWYM. Thanks also to Barbara Grosz, Richard Power, Wheeler Ruml, Donia Scott, and Kees van Deemter for helpful suggestions.

## References

- Baldwin, B. 1997. CogNIAC: high precision coreference with limited knowledge and linguistic resources. In *Proceedings of ACL-EACL Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, 38-45.
- Bouayad-Agha, N., Power, R., Scott, D., and Belz, A. 2002. PILLS: multilingual generation of medical information documents with overlapping content. In *Proceedings of LREC*.
- Clark, H. 1977. Bridging. In *Thinking: Readings in Cognitive Science*: Cambridge Univ. Press, NY.
- Cristea, D., Ide, N., and Romary, L. 2000. Veins theory: a model of global discourse cohesion and coherence. In *Proceedings of ACL-COLING-98*.
- GM. 1980-2002. GM Service Operations Web Pages at URL <http://service.gm.com>.
- Grosz, B. 1977. The representation and use of focus in dialogue understanding. PhD thesis, Univ. of California, Berkeley.
- Grosz, B., Joshi, A., and Weinstein, S. 1995. Centering: a framework for modeling the local coherence of discourse. In *Computational Linguistics*, 21(2): 203-225.
- Grosz, B., and Sidner, C. 1986. Attention, intentions, and the structure of discourse. In *Computational Linguistics*, 12(3): 175-204.
- Johnson, J. 2000. *GUI Bloopers: Don'ts and Do's for Software Developers and Web Designers*. Morgan Kaufmann Publishers, Boston.
- Kibble, R. and Power, R. 2000. An integrated framework for text planning and pronominalisation. In *Proceedings of INLG'2000*.
- Power, R. 1999. Generating referring expressions with a unification grammar. In *Proceedings of EWNLG-99*, 1-9.
- Power, R., Scott, D., and Evans, R. 1998. What you see is what you meant: direct knowledge editing with natural language feedback. In *Proceedings of ECAI-98*, 675-681.
- Shneiderman, B. 1992. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*: 2<sup>nd</sup> Edition. Addison-Wesley, Reading.
- Van Deemter, K. and Power, R. 1998. Coreference in knowledge editing. In *Proceedings of ACL-COLING-98 Workshop on the Computational Treatment of Nominals*, 56-60.
- Walker, M. 2000. Toward a model of the interaction of centering with global discourse structure. In *Verbum*, 22.