A Case Study of Natural Language Customisation: The Practical Effects of World Knowledge

Marilyn A. Walker
lyn@linc.cis.upenn.edu

Andrew L. Nelson
aln@hplb.hpl.hp.com

Phil Stenton
sps@hplb.hpl.hp.com

University of Pennsylvania
Computer Science Dept.
200 S. 33rd St.
Philadelphia, PA 19104

Hewlett Packard Laboratories
Personal Systems Lab
Filton Rd., Stoke Gifford
Bristol, BS12 6QZ, U.K.

Abstract

This paper proposes a methodology for the customisation of natural language interfaces to information retrieval applications. We report a field study in which we tested this methodology by customising a commercially available natural language system to a large database of sales and marketing information. We note that it was difficult to tailor the common sense reasoning capabilities of the particular system we used to our application. This study validates aspects of the suggested methodology as well as providing insights that should inform the design of natural language systems for this class of applications.

1 Introduction

It is commonly accepted that *we understand discourse so well because we know so much*[5]. Hobbs identifies two central research problems in understanding how people interpret discourse. We must characterise: (1) the knowledge that people have, and (2) the processes they use to deploy that knowledge. This includes specifying and constraining the inferential and retrieval processes that operate on what is known[7]. This problem is of practical interest for the design of various types of natural language interfaces (NLI's) that make use of different knowledge sources.

The knowledge used by an NLI is often split into two types. DOMAIN-INDEPENDENT knowledge consists of grammatical rules and lexical definitions. It also includes knowledge used for common-sense reasoning[6]. DOMAIN-DEPENDENT knowledge centres on modeling processes unique to the application task, or the particular relations in the application database. The process of customising an NLI consists in adding the domain-dependent knowledge about a particular application to the domain-independent knowledge that comes with the NLI[4]. Very little has been written about how this customisation is done.

This paper results from a particular customisation effort in which we took a commercially available NLI and attempted to customise it to a large sales and marketing information database installed at a customer site. The application was information retrieval for decision support. We suggest a particular method to be used in the customisation process and evaluate the success of this method. We note a number of problems with using the domain independent knowledge provided with the NLI for our particular application. We also note cases where the inferential processes supported by the NLI do not appear to be appropriately constrained. The application of this method leads to some general results about the process of customisation, as well as some specific insights regarding this type of application and the evaluation of an NLI. Section 2 describes the particular NLI and the application. Sections 3, 4, 5, 6 and 7 describe the methodology that we applied in our customisation effort. Section 8 describes the results of testing the customisation. Finally, Section 9 provides suggestions for customisers or designers of NLI's.

2 NLI and Sales Application

The database was a large and complex on-line sales database, containing information about orders, deliveries, brands, customer preferences, sales territories, promotions and competitors. There were 20-30 different types of records with over 200 views ranging over data summaries of 2-3 years.

Our user group consisted of 50 managers, composed of accounts, brands, commercial and marketing managers, each with different data requirements. They fit the user profile recommended for NLI's[8]. They were relatively infrequent computer users, who were experts in the domain with at least one year's experience. None knew anything about database languages. Some of them had used a previously installed NLI, Intellect, as well as a menu-based interface that accessed

the same set of data[1]. They required ad hoc access to information that was difficult to support with standard reports.

The NLI we worked with was considered state of the art. It appeared to use a pipeline architecture consisting of morphological analysis, parser, semantic interpretation, and database query translator. The semantic representation language was a hybrid of a semantic network and first order predicate logic, which supported time dependent facts, quantified statements, tense information and general sets[3]. In addition, this NLI included a Generator that produced English from the semantic representation language, a Deductive System that reasoned about statements in the representation language using forward and backward chaining, and which handled quantification, time dependent facts and truth maintenance. Among the knowledge sources that came with the NLI was a Dictionary of 10000 initial English words, and a set of Concepts that provided internal notions of predicates, and set and membership hierarchies.

The semantic representation, concepts, and dictionary modules supported both intensional and extensional representation of data. In addition, users could add both new concepts and inference rules to the system with simple declarative sentences.

3 Method

Information sources for the customisation included: the customisation manual, database schema, NL transcripts of users accessing the data in the database using the previous NLI, Intellect, and a test suite of English sentences[2].

Our customisation method had four parts:

1. NL transcript analysis
2. Mapping NL terms onto an Entity-Relation (E-R) diagram
3. Constructing the customisation files
4. Generating a test suite and testing the customisation

We restricted our efforts to implementing and testing coverage of a sub-part of the domain identified as important through analysis of the NL transcripts, namely the deliveries subdomain[2]. The important concepts are listed below and highlighted in Figure 1.

- The Product Hierarchy : Markets, Sectors, Brands, etc.
- The Customer Hierarchy : Corporations, Trading Companies, Concerns
- The Time Hierarchy: Years, Book Months, Book Weeks
- Deliveries of Products to Customers over Time

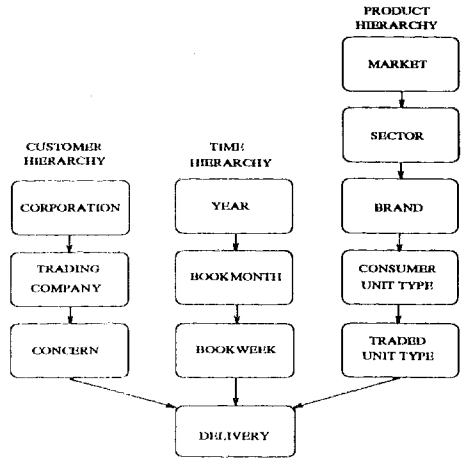

Figure 1: Simplified model of the Sales domain

The following sections will discuss each aspect of the customisation procedure and the issues raised by each step of the method.

4 Analysing the transcripts

The NL transcripts consisted of every interaction with the previous NLI, Intellect, over a period of a year, by our user group, accessing our target database. A detailed account of the transcript analysis can be found in [9]. Here we focus on how the results affected the rest of the procedure.

The transcripts showed that the most important set of user queries were those about deliveries of the different levels of the product hierarchy to the different levels of the customer hierarchy. The transcripts also showed that over 30% of user errors were synonym errors or resulted from the use of terms to refer to concepts that were calculable from information in the database. We collected a list of all the unknown word errors from the Intellect installation. For example using the term *wholesalers* resulted in an Intellect error, but it refers to a subset of trading companies with trade category of WSL. We didn't feel that the syntax of the transcripts was important since it reflected a degree of accommodation to Intellect, but the Intellect lexicon and the unknown word errors gave us a good basis for the required lexical and conceptual coverage. In the absence of such information, a method to acquire it, such as Wizard of Oz studies, would be necessary[10, 1].

[1] In [9] we compare the menu system to Intellect.

[2] The customising team comprised two computational linguists, a computer scientist and two psychologists.

5 Mapping NL terms onto an E-R diagram

The steps we applied in this part of the proposed method are: (1) take the E-R diagram provided by the database designer at the customer site as a conceptual representation of the domain, (2) associate each lexical item from the transcript analysis with either an entity or a relation, (3) Refine and expand the E-R diagram as necessary.

We started with a list of lexical items e.g. *markets, sectors, brands, deliver, pack size, date, corporate, trading concern, customer location,* that were part of the Intellect lexicon or had appeared in the transcripts as unknown words. By placing these lexical items on the E-R diagram we were able to sketch out the mapping between user terms and database concepts before committing anything to the customisation files[3]. However, we found mapping vocabulary onto the E-R diagram to be more difficult than we had anticipated.

First, a number of words were ambiguous in that they could go in two different places on the E-R diagram, and thus apparently refer to multiple concepts in the domain. This was most clearly demonstrated with certain generic terms such as *customer*. *Customer* can be used to refer to a relation at any level of the customer hierarchy, the *concern*, the *trading company* or the *corporation*. It can also be associated with the attribute of the *customer reference number* which is a key value in the 'Concern' database relation.

Second, some words were based on relationships between two entities, so they could have gone in two places. For instance *market share* is calculated from information associated with both the *market* entity and with the *trade sector* entity. Similarly, the term *delivery* refers to a relation between any level of the product hierarchy and any level of the customer hierarchy. Yet there was no entity that corresponded to a delivery, even though it was one of the main concepts in the domain.

In both of these cases we created new entities to refer to concepts such as *delivery* and *market share*. We were then able to indicate links between these concepts and other related concepts in the domain and could annotate these concepts with the relevant vocabulary items. In some cases it was difficult to determine whether a term should be a new entity. For instance the term *wholesalers* refers to members of the *trading company* entity with a particular value in the trade category attribute. However since trade category is not used in any other relation, it doesn't have a separate entity of its own. In this case we left *wholesaler* as a term associated with *trading company*.

Third, in our lexicon there were operators or predicators such as *less than, greater than, equal to, at least, change, decrease, latest estimate, over time, chart, graph, pie, during, without, across, display, earliest, available*. These operators were domain independent operators; some of them were synonyms for functions that the system did support. Since these seem to be concepts perhaps related to the task, but not specific to the domain, for convenience we created a pseudo entity on the E-R diagram having to do with output and display concepts such as graphing, ranking, displaying information as a percentage etc.

Finally, there were also terms for which there was no database information such as *ingredients* and *journey*, ambiguous terms such as *take, get, accept, use*, as well as terms that were about the database itself, such as *database, information*. For other terms such as *earliest* or *available* it was difficult to determine what domain concepts they should be associated with.

However, the benefits of this method were that once we had made the extensions to the E-R diagram, then all synonyms were clearly associated with the entities they referred to, words that could ambiguously refer to multiple concepts were obvious, and words for which a calculation had to be specified were apparent. We were also able to identify which concepts users had tried to access which were not present in the domain. Once this was done the customisation files were built incrementally over the restricted domain.

6 Constructing the customisation files

The input to this part of the process was the annotated E-R diagram as well as the test suite. We chose not to use the menu system customisation tool that was part of the NLI[4]. We preferred to use an interface in which declarative forms are specified in a file.

As we developed the customisation file incrementally over the domain, we ensured that all the synonyms for a concept were specified, and thoroughly tested the system with each addition. This section discusses constructing the customisation file. In section 7, we discuss the test suite itself. The results are discussed in section 8.

6.1 Grammatical and Conceptual Information

The customiser's job is to link domain dependent knowledge about the application to domain independent knowledge about language and the world. Constructing a customisation file consisted of specifying a number of forms that would allow the NLI to pro-

[3] In a perfect world, the NLI would target an E-R diagram and the mapping from the E-R diagram to the database would be an independent aspect of the semantic modelling of the domain.

[4] The menu system was very large and unwieldy with many levels, too many choices at each level, and a lack of clarity about the ramifications of the choices.

duce a mapping between English words, database relations, attributes and values, and concepts used in common sense reasoning by the deductive component of the NLI.

A database relation, such as 'Deliveries', could have nouns or verbs associated with it, e.g. *delivery* or *deliver*. In the case of verbs, mappings are specified to indicate which attributes correspond to each argument slot of the verb.

In either case, both relation and attribute mappings, give one an opportunity to state that the relation or the attribute is a particular type of entity. This type information means that each concept has type preferences associated with its arguments. The NLI provided types such as `person`, `organisation`, `location`, `manufactured object`, `category`, `transaction`, `date` or `time duration`. The specification of these types supplies background information to support various inferential processes. There are three types of inference that will concern us here:

- Coercion
- Generalisation and Specification
- Ambiguity resolution

COERCIONS depend on the type information associated with the arguments to verbs. For example, consider a verb like *supply* with arguments *supplier* and *suppliee*. Let's say that suppliers are specified to be of type `concern`, and suppliees are of type `project`. Then the query *Who supplied London?* violates a type preference specified in the customisation file, namely that suppliee is a project. A coercion inference can coerce London, a `city`, to `project`, by using the inference path [`project` located `location` in `city`]. Then the question can be understood to mean *who supplies projects which are in London?*[3].

GENERALISATION inferences can support the inference that *Lite is a kind of Cheese* given other facts such as *Lite is in sector Full Fat Soft* and *Full Fat Soft is a kind of Cheese*. A similar inference is supported by the type `organisation` ; if X works for organisation Y, and Y is a suborganisation of organisation Z, then the NLI is supposed to be able to infer that X works for Z.

AMBIGUITY resolution consists of filling in underspecified relations. A common case of unspecified relations are those that hold between the nouns of noun-noun compounds (n-n-relations). For example a *motorola processor* is a processor with motorola as the manufacturer. A *department manager* is a manager of department. The specification of conceptual types in the customisation file is intended to support the inference of these unspecified n-n-relations. For example, the NLI first interprets these with a generic *have* relation and then attempts to use the conceptual types to infer what relation the user must have intended. Similarly from the knowledge that an attribute is a `location`, the NLI can infer that it can be used as an answer to a question about where something is.

6.2 Difficulties

A minor difficulty in developing the customisation file was that we identified lexical items for which there was no information in the database. In this case we used a facility of the NLI by which we could associate helpful error messages with the use of particular lexical items. In cases where the concept could be calculated from other database information, we were able to use the NLI to extend the database schema and specify the calculations that were needed in order to support user's access to these concepts.

The more major difficulty was to determine which of the concepts that the NLI knew about, was the type to use for a specific domain lexical item. For example in specifying the 'Markets' database relation, target phrases might be *the chocolate market, the market chocolate, sales of chocolates, how much chocolate* or *kinds of chocolate*. One of the types available was `category` which seems to be the way the key marketname is used in the phrase *the chocolate market*[5]. However, another option was to create an attribute mapping for marketname. Attribute mappings can specify that an attribute type is one of a different set of types such as a `unique identifier`, `a name`, `a pay`, `the employer`, or a `superorganisation`. And some of these have subtypes, e.g. name can be of type proper, classifier, common, model or patternnumber. So perhaps if one wants to say *sales of chocolates* then marketname should be a common name. A solution would be to say marketname belongs to a number of these types, possibly at the expense of overgenerating. In the case of this particular NLI, attempting to do this generated warnings.

7 Generating the test suite

The test suite of sentences was constructed by selecting sentences that cover the requirements identified by our transcript analysis from the published test suite [2]. We then substituted concepts to reflect our subdomain of sales. Sentences were generalised across hierarchies in the domain and with respect to various words for relations in a hierarchy (e.g. *are in, belong to, contain, have, are part of, are kind of*).

As soon as we began testing our first customisation file mappings, it was immediately obvious that this test suite was inappropriate for use in early customisation. This was because it was partitioned with respect to

[5]The documentation on a category says that objects "fall into" categories. If C is a category you can ask, *"who fell into C ?"* It is not clear as to whether this meant that 'Markets' was a category.

syntactic form and not with respect to the boundaries of customisation sub-domains. This is a common feature of most test suites. It also contained some queries which had too much syntactic complexity to be of use in identifying separable problems in the customisation file.

We therefore created a smaller set of deliveries test queries that used only the more simple syntactic forms and which was organised with incremental domain coverage. This was ideal for iterative development of the customisation, and enabled us to concentrate on getting the basic coverage working first. Later in the customisation we used the more complete syntax-based test suite to get a more complete picture of the limitations of the resulting system with respect to user requirements. We will discuss a possible remedy to the situation of having two distinct test suites in the conclusion.

8 Testing the customisation

Some of the coverage limitations were specific to this NLI, but there are some general lessons to be learned. Many of the pernicious problems had to do with the NLI's ambitious use of common-sense knowledge. This section briefly discusses some of the limitations in syntactic coverage that we detected. The remainder of the discussion focusses on the NLI's use of common sense reasoning.

8.1 Testing syntactic coverage

While the syntactic coverage of the NLI appeared to be better than the Intellect system, we were able to identify some coverage limitations of the system.

NUMERIC QUANTITIES like the number of 'cases' delivered and number of tonnes delivered were difficult to handle. We managed to engineer coverage for *How many* queries concerning the number of cases of products, but were unable to get any coverage for *How much* queries concerning number of tonnes.

COORDINATION worked for some cases and not for others with no clear dividing line. Switching the order of noun conjuncts, e.g. in *List the market and sector of Lite*, could change whether or not the system was able to provide a reasonable answer. Similarly NEGATION worked in some cases and not in others that were minimally different. It appeared that the verb and some of its arguments could be negated *What was not delivered to Lee's?*, while others could not, *What was not delivered in January.*

DISCOURSE related functionality, such as interpreting pronouns and the use of ellipsis was also variable at best, with further refinements to previous queries such as *and their sales* not properly interpreted.

8.2 The effects of world knowledge

A number of problems concerned the set of predefined concepts that came with the NLI, and that that were used in the customisation file as types for each lexical item and its arguments. These seemed to be domain independent concepts, but to our surprise we discovered that this representation of common-sense knowledge incorporated a particular model of the world. For instance, a lot of support was provided for the concepts of **time** and **time durations**, but time was fixed to the calendar year. Our domain had its own notion of time in terms of bookweeks and bookmonths in which weeks did not run from Sunday to Sunday and months could consist of either 4 or 5 weeks. The English expression *weekly deliveries* was based on this and manager's commissions were calculated over these time durations.

There were a number of cases where domain dependent knowledge was embedded in the presumably domain independent conceptual and dictionary structure of the NLI. For instance *how much* was hard-wired to return an answer in dollars. The point is not that it didn't respond in pounds sterling, but rather that our users wanted amounts such as cases, tonnes, and case equivalents in response to questions such as *How much caviar was delivered to TinyGourmet?*

Another feature of world knowledge which made customisation difficult was the fact that predefined concepts comprise a set of built-in definitions for certain words. These definitions were part of the core lexicon of 10,000 words provided with the system, but the customiser is not given a list of what these words are[6]. This causes mysterious conflicts to arise with domain-specific definitions. For instance, we had to first discover by careful sleuthing that the system had its own definitions of *consumer, customer, warehouse, sale,* and *configuration,* and then purge these definitions. It was not possible to determine the effects of these purges in terms of other concepts in the system.

In particular, there were concepts that were not easy to remove by purging lexical definitions such as the concept of TIME mentioned above. The ambiguity of predefined concepts also arose for certain verbs. For example, the verb *to have* was pre-defined with special properties, but no explicit definition was made available to the customiser. It was impossible to determine the effects of using it, and yet it seemed unwise to purge it.

Our application had a great need for GENERALISATION type inferences due to the product, customer and time hierarchies (see figure 1). The most common verb was *deliver* and this could refer to *deliveries* of any level in the product hierarchy to any level in the customer hierarchy. We spent a great deal of time trying to get this to work properly and were not able to. In the examples below (Q) is the original query, (P) is the paraphrase provided by the system and (R) is the

[6] Presumably because this is consider proprietary knowledge.

system's response. In example 1 the customer *Lee* is at the level of *corporation* and the query is properly interpreted, resulting in a table of product, customer, delivery date, etc.

(1) Q: What are the sales of Krunchy in Lee?

P: List Lee's Krunchy sales.

However, in 2 the customer *Foodmart* is at the level of *trading company* and a query with the identical syntactic form is interpreted completely differently.

(2) Q: What are the sales of Krunchy in Foodmart?

P: List the Krunchy in Foodmart sales.

R: There aren't any brands named Foodmart.

Other problems were not so clearly problems with common sense knowledge but rather with inappropriately constrained inferential powers. Some of these were best identified by examining the paraphrases that the generator produced of the semantic interpretation of a user query or statement. By the paraphrase provided in (3)P, it appears that the n-n-relation in *report groups* has been interpreted as *have*.

(3) Q: What do you know about report groups?

P: What do you know about groups that have REPORT?

R: The database contains no information about which groups have customers.

Then another default inference is made, which consists of assuming that an unknown proper noun is of type *customer*. This results in the response given in (3)R. Of course to the user, this response seems to indicate that the system has not at all understood his query.

Another example of a non-fruitful assumption of a *have* relation for a non-specified n-n-relation can be seen in (4)R below. The NLI first expands the proper name BSL to traded unit type BSL, then apparently treats this as a noun noun compound with an unspecified n-n-relation. This relation is then filled in with the *have* relation which appears in (4)R.

(4) Q: Show the total sales of bsl, bsj and bsr to Lee's PLC.

P: List the total sale of traded unit type BSL, the total sale of traded unit type BSJ and the total sales of traded unit type BSR to Lee's PLC.

R: Traded unit types don't have traded unit types. Consumer unit types have traded unit types.

In example (5), the NLI appears to make an unwarranted inference that the number 17 must refer to 17 dollars. It also fills in a null noun, taking the sentence to actually mean *how much N was krunchy sold*.... It replaces this null noun with the type *traded unit type* which is given as a default for how much queries.

(5) Q: how much was krunchy sold between week 17 in 1988 and week 52 in 1988?

P: How much traded unit type that was sold to krunchy costs between 17 and 52 dollars?

R: The database contains no information about how expensive traded unit types are.

It seems that the semantic information that the system has, such as knowing that krunchy is a brand and that sales are of a product to a customer, should let it overcome the slightly nonstandard syntax *how much was krunchy sold*. However it apparently pays more attention to that aspect of the syntax here, while ignoring the fact that 17 is specified to be a designator of a book week.

9 Conclusions

The NL transcript analysis proved useful to identify the target coverage and to focus our experiment on a priority part of the domain. In most cases transcript information will not be available and so interview data or experimental Wizard-of-Oz data[10] will have to be generated to make explicit the users' models of the domain.

The E-R model of the domain was very useful for carrying out an incremental development of the customisation file. It lets the customiser know where the reasonable domain boundaries lie, in order that subparts of the customisation can sensibly be developed and tested in isolation. In addition the customisation was simplified by having the entities and attributes of the E-R model labelled with the domain vocabulary in advance. Thus the process of associating synonyms with appropriate customisation file relations and attributes was straightforward.

The main limitation of the approach seem to be that E-R diagrams are too limited to capture the use of the vocabulary in the domain. We used an E-R diagram because it was the conceptual representation available for the domain and because it is the most prevalent semantic modeling tool used in database design. However, it does not in fact allow one to represent the information that one would like to represent for the purposes of linking NL concepts and lexical items to the domain. The only semantic information associated with relations that is represented in an E-R diagram are whether they are many-to-one or one-to-one. The attributes of the entity that participate in the relation are not indicated specifically. The representation

should be much richer, possibly incorporating semantic concepts such as whether a relation is transitive, or other concepts such as that an attribute represents a part of a whole. Of course this is part of what the NLI was attempting to provide with its concept hierarchy and dictionary of 10000 initial words.

But it seemed that one of the main difficulties with the NLI was in fact exactly in attempting to provide a richer semantic model with common sense information to support inference. This is commonly believed to be helpful for the portability of a NL system across a number of domains. We found it a hindrance more than a help. Some predefined concepts had to be purged from the lexicon. Some definitions were difficult to delete or work around e.g. time definitions. The problems we encountered made us wonder whether there is any general world knowledge or whether it is always flavoured by the perspective of the knowledge base designers and the domains they had in mind.

The process was not helped by the black box nature of the NL system. The general problem with black box systems is that it is difficult for a customiser to get an internal model of system. It would help a great deal if the world model was made available directly to the customiser, the semantics of each concept was clearly defined, and a way to modify rather than purge certain parts of the conceptual structure was made available. The customiser should not be left to learn by example.

During customisation of the NL system we found our user requirements test suite difficult to use for debugging purposes. The test suite had to be modified to reflect concepts in the database rather than syntax. This is because customisations must be done incrementally and tested at each phase. A solution to this problem is first to ensure that the test suite has a number of sentences which test only a single syntactic construction. Second, store the test suite components in a database. Each component would be retrievable through the semantic class it belonged to (i.e Temporal Expression or Complex NP). In addition each component would be retrievable through the concepts of the E-R diagram that it accessed. Then it should be possible to generate test suites that are usable by developers for the purpose of testing customisation files. Simple queries of the test suite database about a particular concept would generate appropriate test sentences whose semantic categories and category fillers were limited to that concept.

References

[1] Nils Dahlback and Arne Jonsson. Empirical studies of discourse representations for natural language interfaces. In *Proc. 4th Conference of the European Chapter of the ACL, Association of Computational Linguistics*, pages 291–298, 1989.

[2] Daniel Flickinger, John Nerbonne, Ivan Sag, and Thomas Wasow. Towards evaluation of nlp systems, 1987. Presented to the 25th Annual Meeting of the Association for Computational Linguistics.

[3] Jerrold M. Ginsparg. A robust portable natural language data base interface. In *Proc. 1st Applied ACL, Association of Computational Linguistics, Santa Monica, Ca.*, pages 25–30, 1983.

[4] Barbara J. Grosz. Team: A transportable natural language interface system. In *Proc. 1st Applied ACL, Association of Computational Linguistics, Santa Monica, Ca.*, 1983.

[5] Jerry R. Hobbs. The logical notation: Ontological promiscuity, chapter 2 of *discourse and inference*. Technical report, SRI International, 333 Ravenswood Ave., Menlo Park, Ca 94025, 1985.

[6] Jerry R. Hobbs, William Croft, Todd Davies, Douglas Edwards, and Kenneth Laws. The tacitus commonsense knowledge base. Technical report, SRI International, 333 Ravenswood Ave., Menlo Park, Ca 94025, 1987.

[7] Aravind K. Joshi and Scott Weinstein. Control of inference: Role of some aspects of discourse structure - centering. In *Proc. International Joint Conference on Artificial Intelligence*, pages pp. 385–387, 1981.

[8] S.R. Petrick. On natural language based computer systems. *IBM Journal of Research and Development*, pages 314–325, July, 1976.

[9] Marilyn Walker and Steve Whittaker. When natural language is better than menus: A field study. Technical Report HPL-BRC-TR-89-020, HP Laboratories, Bristol, England, 1989.

[10] Steve Whittaker and Phil Stenton. User studies and the design of natural language systems. In *Proc. 4th Conference of the European Chapter of the ACL, Association of Computational Linguistics*, pages 116–123, 1989.