Transportable English-language processing for office environments

by BRUCE W. BALLARD, JOHN C. LUSTH, and NANCY L. TINKHAM

Duke University
Durham, North Carolina

ABSTRACT

This article describes the Layered Domain Class system (LDC), a state-of-the-art natural language processor whose major goals are (1) to provide English-language retrieval capabilities for medium-sized office domains that have been stored on the computer as text-edited files, rather than more restrictive database structures and (2) to eliminate the need to call in the system designer when extensions into new domains are desired, without sacrificing the depth or reliability of the interface. Early developments in the design of portions of LDC were presented at NCC-83, and the entire system became operational in July 1983. The article gives an overview of the construction of the system, gives examples of the English structures provided for, briefly describes the most recently completed portions of the system, and mentions current directions the project is taking.
INTRODUCTION

During the 1970s, a number of experimental systems providing limited natural language processing capabilities were developed to permit computer access by casual or untrained users. The most frequent application was for database query, and other application areas have included automatic programming, computer-aided instruction, office automation, and medical information retrieval. Several prototype systems have been tested with prospective users, and at least one system (INTELLECT) has been used in several dozen commercial database environments and is currently being marketed by IBM.

Our interest is in adapting and extending techniques developed for previous natural language (NL) systems, especially those used in database query systems and in our own natural language programming system NLC\textsuperscript{1,2,3} for use in office environments. This article gives a brief overview of the Layered Domain Class system (LDC), a state-of-the-art natural language processor whose primary goals are (1) to provide English-language retrieval capabilities for medium-sized office domains that have been stored on the computer as text files, that is, files produced with a standard text editor, rather than more restrictive database structures, and (2) to eliminate the need to call in the system designer when extensions into new domains are desired, without sacrificing the depth or reliability of the interface. Depth of an English-language processor refers to the degree to which the system supports the natural syntax and semantics of the language. That is, we distinguish a natural language system from English-like languages that make use of English vocabulary in what otherwise operates as a formal language interface.

In designing LDC, we have sought to identify a broad class of domains that have similar structure but contain entirely different sorts of data. For the prototype LDC system, we have chosen to consider domains with semantics similar to those of our previous NLC matrix-domain system. Some of the more abstract properties we have incorporated are hierarchical decomposition, uniform breakdown of entities, and implicit orderings of domain elements. Thus, LDC provides capabilities to learn about domains where decomposition serves as the primary structuring relation. We refer to these as layered domains. Previous papers discuss some of the mathematical and psychological properties of these domains\textsuperscript{4} and give partial descriptions of pre-prototype system components.\textsuperscript{5} A discussion of the internal processing that takes place during the processing of inputs by LDC can also be found elsewhere.\textsuperscript{6}

OVERVIEW OF LDC

An overview of the environment in which LDC operates is suggested in Figure 1. As shown in Figure 1, our system is designed to take as input preexisting data files that will have been created using a standard text editor, and LDC is composed of three major modules. The first of these is the preprocessor, through which an experienced user, or “super-user,” customizes the system to operate in a new domain. As a result of preprocessing, various files are created that provide domain-specific information for later processing. The next module is the English-language processor, which receives English inputs, currently in typed mode, and by a series of steps to be described later produces an appropriate formal query for the third module of LDC, the retrieval module. As shown in Figure 1, our retrieval module has been designed to be usable in stand-alone mode, independent of the English processing portion of LDC, somewhat like a conventional database retrieval module. We will occasionally refer to the English-language processor and retrieval module collectively as the User-Phase processor.

KNOWLEDGE ACQUISITION

The initial interaction between a user and LDC, which involves telling the system about a new domain, consists of a dialogue with the preprocessor, which we call “Prep.” Prep operates by acquiring information about the names of each type of entity of the domain; the nature of the relationships among them; the English words that will be used as nouns, verbs, and modifiers; morphological and semantic properties of these new words; and the relation between the conceptual domain structure and the physical objects of the raw data file.
For example, in describing a data file that contains information about building locations, the user might say the following:

1. A "room" is an independent domain object.
2. A given room is found on exactly one "floor."
3. An "office" and a "conference room" are types of rooms, and rooms may be spoken of as "large," "vacant," and "small."
4. A floor is said to be "restricted" if it contains one or more offices.
5. Information about the location of a room is found in the Loc column of some particular text file.

In addition to its primary role of asking for information, Prep also allows the user to probe its knowledge and make corrections or updates as desired.

THE ENGLISH-LANGUAGE PROCESSOR

In this section, we seek to convey a feeling for the types of English inputs LDC is able to process. Our initial interest in developing LDC was to study the specification by users of complex semantics; therefore, we chose for our system to expect noun phrases rather than full question forms. We note that the power of the system is only minimally reduced by restricting users to noun phrases because there is a corresponding equivalent noun phrase for most questions. For example, the answer to the question

What grade did Mary get from Biermann?

is precisely the referent of the noun phrase

the grade that Mary got from Biermann

We first discuss accepted forms, then give examples of presently unaccepted forms.

English Structures Processed by LDC

Noun phrases in LDC consist of two types: (1) proper-noun phrases, such as "Jack," "CPS201," and "a B +," and (2) descriptive phrases, such as "the best student Ballard taught in CPS201." Because the syntax of proper-noun phrases is trivial, the following presentation deals with descriptive phrases. It is important to note that the presence of prepositional phrases, comparative phrases, and relative clauses leads to nesting of one or more noun phrases within another.

Descriptive noun phrases are composed of a head noun preceded by zero or more premodifiers, usually single words, and followed by zero or more postmodifiers, usually multiple-word phrases. Permissible premodifiers in the current LDC grammar are the articles the, a, and an; ordinal numbers; superlatives; adjectives; noun modifiers; and single-word possessives. Some examples are:

article: the offices
ordinal: the third floor

superlative: the largest room
adjective: the vacant offices
noun modifier: the conference rooms
possessive: Biermann's lab

Cardinal numbers may occur in a noun phrase if they appear together with ordinals or superlatives:

the first two floors

Premodifiers may be used freely in combination with one another:

the largest vacant office

However, there are restrictions in English regarding the usage of premodifiers with one another, the ordering of premodifiers, and the choice of modifiers for nouns. These restrictions are upheld in the LDC grammar so that constructions such as the following are disallowed:

the smallest last room
largest Ballard's office
the lastly person

It is important that such spurious constructions be disallowed in order to help reduce potential ambiguities of nested structures, problems caused by typing errors, or problems of the noise present in spoken input.

The simplest form of postmodifier provided for in LDC is the predicative specification of an ordinal:

section 3 of CPS51

The simplest multiple-word postmodifier is the prepositional phrase, which consists of a preposition followed by an arbitrarily complex noun phrase:

the undergraduates in the course Joe failed
the student with the lowest grade in EE291

LDC also provides a variety of relative clause structures. For example, the system accepts all of the following noun phrases derived from the sentence "Ballard gave a B to Nancy."

the professor who gave a B to Nancy
the professor who gave Nancy a B
the professor by whom Nancy was given a B
the professor whom Nancy was given a B by
the grade that Ballard gave to Nancy
the grade Ballard gave to Nancy
the grade that Ballard gave Nancy
the grade Ballard gave Nancy
the grade which was given to Nancy by Ballard
the grade given to Nancy by Ballard
the grade given Nancy by Ballard
the student to whom Ballard gave a B
the student Ballard gave a B to
the student who was given a B by Ballard
the student given a B by Ballard

For simplicity we have shown these 16 forms with only proper-noun embedded phrases, but in general arbitrary noun phrases may occur, as in:

instructors who gave an F to a student who made a passing grade in a course taught by Ballard

The words which and that may be substituted for each other in the sentences shown. A relative clause will occasionally contain a verb with a particle, such as add up or give up, and the LDC grammar allows the particle to occur either before or after the object of the verb:

the students who made up a graduate course
the students who made EE157 up

Finally, LDC accepts certain comparative structures. One such type of construction is a relative clause containing the comparative form of an adjective and optionally containing a form of the verb to be:

the courses that were smaller than CPS152 (was)
the grades lower than B
the courses smaller than Ballard’s smallest course

A second type of comparative construction in LDC is somewhat different in that it functions as a noun phrase. This particular form also extends to superlatives:

the larger of CPS200 and CPS51
the largest of CPS200, CPS51, CPS224, and EE157
the largest of CPS215, Carroll’s courses, and EE209
the largest of the courses Anne took

**English Forms Not Presently Accepted**

To give an idea of the limits of LDC, we will list some of the constructions that cannot be processed at this time. First, LDC does not allow the use of cardinal numbers without an ordinal or superlative, as in:

the professor who failed six students

Second, LDC is not able to derive the meaning of a participial adjective (for example, passing) automatically from the meaning of the verb that is its root (for example, pass); at present, “-ing” forms of verbs must be included separately in the dictionary and labeled as adjectives. This is a limitation of the parser and is transparent to the user. Third, LDC is not yet able to parse “discontinuous constituents,” constructions such as:

add the positive entries up in row 3
a higher grade than John made

in which components of a single phrase or clause have been separated by another sentence element. Fourth, LDC does not currently allow arbitrary nouns to be used as modifiers, as in “the B students,” due to the difficulty in determining the intended meaning. Fifth, for similar reasons, the system does not yet handle possessive phrases such as “the best EE157 student’s instructor,” consisting of a possessive noun with premodifiers that function as premodifiers for another noun. Finally, LDC is not yet able to handle pronoun references—personal pronouns, demonstrative pronouns and determiners, and words such as each and all when used as pronouns—because it is not yet able to use context to determine the referents of pronouns.

**THE RETRIEVAL MODULE**

The retrieval module of LDC has been designed to meet several criteria:

1. To be able to access loosely structured text files of the kind typically maintained in office environments rather than more formal database structures
2. To provide a rich repertoire of primitive operations
3. To provide a macro facility for user customizations so that frequent compositions of primitives can be made in abbreviated form
4. To be able to deal with many user domains without intervention on the part of the system designers
5. To render query syntax independent of the specific physical structure of the data file being accessed

In addition to these criteria, the retrieval component is expected to be useful both in stand-alone mode and as a convenient retrieval component for LDC.

The query language supported by our retrieval module is very much like formal query languages for database query, but there are some important differences. For example, our provision for macros has no counterpart in most conventional systems. Furthermore, like many modern programming languages, our query structures make no distinction among levels of operations, and any sequence of commands can occur as an

From the collection of the Computer History Museum (www.computerhistory.org)
embedded query inside any of the others, wherever a single primitive value is required. We also provide an ordinal retrieval function and a percentage informational function that are nonstandard.

DATA FILES PROVIDED FOR

It is convenient to regard LDC as viewing its text-edited input file as a sequentially accessed file in which each line corresponds to a separate record. As a familiar example, which we will use for much of the remainder of the article, consider a final grades domain, with text lines such as:

```
CPS51.2 Ballard Young, Charles A-
CPS241.1 Starmer Smith, John B+
CPS241.1 Starmer Taylor, Sue A-
```

Although a certain degree of time and space overhead may result from some of the text-edited files LDC allows, most of the domains for which our interface has been designed are on the order of hundreds of records, not hundreds of thousands of records, so time and space concerns are less critical than for large conventional databases. Finally, we note that LDC makes a clear distinction between the conceptual and the physical organization of its data file, thereby allowing text-edited files to be more loosely structured than most formal database structures.

FUTURE WORK

We have described a fully operational NL processor that reached the prototype stage in May 1983. Some of the features currently being worked on are negation, limited conjunction, more elaborate verb forms, and capabilities for multiple files. Several additional capabilities we would like to provide for were mentioned earlier, and several of these are also being worked on. In the case of pronouns, we expect to adapt the domain-independent strategies developed for NLC based on a “focus list” concept similar to that being used in the related NL efforts at Duke. 7 We are also engaged in restructuring Prep, the knowledge acquisition component of LDC, to permit more English-like, as opposed to formal, specifications. Another important direction we are considering is the incorporation of the voice processing techniques being used in Biermann’s VNLC and VIPS systems. 7,8

The implementation of negation is virtually complete; therefore, we shall briefly mention how it is being handled. First of all, postnominal modifiers may be negated by using the word “not,” as in:

- students not in CPS241
- CPS215 students who made a grade not higher than a B
courses that Steve did not take
- instructors who did not give an F to Bill

It is instructive to note the inherent ambiguity of English phrases such as “students not failed by Ballard,” which might or might not be intended to include students not taught by Ballard. Our system in fact returns this broader interpretation, as the user can obtain the narrower meaning by asking for “Ballard’s students not failed by Ballard.” Certain prenominal modifiers may also be negated by using “non,” as in:

- a non passing grade
- the non graduate students in Starmer’s course

Clearly, these facilities for negation are somewhat awkward when compared to the rest of the English structures of LDC, but the feature is a semantically important one. When the intended generality of negative semantics has been achieved, attention will be given to making the feature more natural.

RELATED WORK

The experimental LDC system is closest in its present form to database interface systems, because of its question-answering behavior. However, our overall research program involves the development of methods whereby complex semantics may be specified by users of an office system, regardless of whether the application of the system is for answering questions, carrying out commands (as in our previous NLC system), or performing some other task. It is therefore appropriate that we mention related efforts to customize NL systems.

The first serious attention to large-scale customizations by users was made by the REL system. 9 Recent work by these researchers at Caltech is represented by the ASK 10 and POL 11 systems, which seek to provide users with access to various software services in addition to providing simple question-answering facilities. Their emphasis is on providing for broad kinds of capabilities, whereas our effort has been to allow very complex specifications for a restricted class of domains.

A major effort seeking to allow for transportability at the database-administrator level is the TEAM project at SRI. 12,13 The TEAM approach is to carry out an interactive dialogue with database administrators; the system asks questions that enable it to acquire a lexicon relating to the language to be used, a conceptual schema telling about the conceptual relations among objects, and a database schema telling about the underlying database format. A system similar to TEAM, also being developed at SRI but more loosely related to conventional database systems than TEAM, is KLAUS. 14

Other current work in transportable NL system design includes a system being designed at Bell Labs, 15 the IRUS system at BBN, and the CONSUL system 17,18 at ISI. The last of these has special potential value for office environments because it is directed toward software services at personal workstations.

ACKNOWLEDGMENTS

The authors wish to thank Alan Biermann, Martha Evens, Gary Hendrix, George Heidorn, Martha Palmer, Frank Star-
mer, Sharon Salveter, and Bozena and Fred Thompson for valuable discussions on our work.

This research has been supported in part by the National Science Foundation, Grant Numbers MCS-81-16607 and IST-83-01994, and in part by the National Library of Medicine, Grant Number LM-07003.

REFERENCES


