Short Communication
Information Retrieval and Language Processing

Interactive Consulting via Natural Language

Stuart C. Shapiro and Stanley C. Kwasny
Indiana University

Interactive programming systems often contain help commands to give the programmer on-line instruction regarding the use of the various systems commands. It is argued that it would be relatively easy to make these help commands significantly more helpful by having them accept requests in natural language. As a demonstration, Weizenbaum's ELIZA program has been provided with a script that turns it into a natural language system consultant.

Key Words and Phrases: interactive programming, time-sharing systems, natural language processing, computer assisted instruction

CR Categories: 3.32, 3.42, 3.79, 4.49

Introduction

Many interactive systems include a mechanism for automatic dissemination of information regarding the use of its commands. Typically, the user gets this information by entering a basic "help" command and providing the name of the command he wants information about. For example, on the DECsystem-10 [3], the user may type HELP, and get information on the HELP commands; HELP*, and get the names of documented features; or HELP ⟨name⟩, and get information on the feature ⟨name⟩. Figure 1 shows the results of typing HELP and HELP* on the system available at Indiana University.

The problem with such help commands is that the user must know which command he wants information about. If, instead, he only knows what he wants to do and wants to find out the proper command to use, he is reduced to a sequence of guessing command names. Help commands should be more user oriented, allowing the user to describe in his own terms what he wants to do. The system would interpret the request and provide information on how to accomplish the desired task.

Interactive systems consultants (help commands) are excellent applications for natural language understanding programs. Since the context which the systems consultant must deal with is limited, even unsophisticated natural language programs are capable of dealing with it. The ease with which such consultants may be programmed and their usefulness argue that large interactive systems be provided with natural language consultants.

Copyright © 1975, Association for Computing Machinery, Inc. General permission to republish, but not for profit, all or part of this material is granted provided that ACM's copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Association for Computing Machinery.

Authors' addresses: S. C. Shapiro, Computer Science Department, Indiana University, Bloomington, IN 47401; S. C Kwasny, Computer and Information Science Department, Ohio State University, Columbus, OH 43210.

A Natural Language Consultant

Lest the reader fear that we are proposing an extensive research project rather than a program well within the state of the art, let us explain the minimal requirements of a natural language understanding system and why the systems consultant is a good application.

We will say that a system understands natural language if a user who knows what the system is capable of but who has not been specifically trained in the system's input language (i.e. its domain of competence) can phrase an input to the system and, possibly after some clarifying dialogue (see, for example, [1]), have his input satisfactorily handled. The sophistication and complexity required of the system depend on its domain of competence. Relatively sophisticated systems have been written to obey commands to manipulate blocks on a tabletop [12] and to retrieve scientific information on lunar rocks [13]. Newell et al. [8] discuss varying de-

Fig. 1. Help on the DECsystem-10.

```
.HELP

HELP COMMAND (12/27/71) ====
THE HELP COMMAND PRINTS OUT HELPFUL DOCUMENTATION ON
VARIOUS SYSTEM FEATURES.  THE COMMAND

 .HELP
WILL PRINT OUT THIS MESSAGE.

.HELP *
WILL PRINT OUT THE NAMES OF ALL CURRENTLY AVAILABLE INFO.

.HELP <NAME>
WILL LOOK FOR, AND PRINT OUT THE INFO ABOUT THE SYSTEM
FEATURE NAMED IN <NAME>, FOR EXAMPLE

.HELP DIRECT
WILL PRINT OUT INFO ON THE DIRECTORY COMMAND.

ONLY THE FIRST 6 CHARACTERS OF THE ARGUMENT ARE
LOOKED AT, THEY MUST BE A-Z, 0-9, OR *.


.HELP *

HELP IS AVAILABLE FOR THE FOLLOWING:
ABACUS  BASIC   BATCON  BLIS10  BOOT11  CDRSTK  COBDDT  COBOL
COBRG   CREF    DELFIL  DIRECT  DSKRAT  DUMPER  FAILSA  FGEN
FILCOM  FORTRA  FUDGE2  GLOB    GRIPE   HELP    IMPORT  ISAM
LIBARY  LINK    LPTSPL  OMOUNT  OPSER   PIP     PLTSPL  PTPSPL
QUEUE   QUOLST  RERUN   SETSRC  SORT    SOUP    SPACE   SPRINT
SYSDPY  SYSERR  TECO    UMOUNT  2741

THE MONITOR HAS THE FOLLOWING COMMANDS:
ASSIGN  ATTACH  BACKSP  CCONTI  CLOSE   COMPIL  CONTIN  COPY
CORE    CPUNCH  CREATE  CREF    CSTART  CTEST   D       DAYTIM
DCORE   DDT     DEASSI  DEBUG   DELETE  DETACH  DIRECT  DISMOU
DSK     DUMP    E       EDIT    EOF     EXECUT  FILE    FINISH
FUDGE   GET     HALT    HELP    INITIA  JCONTI  KJOB    LABEL
LIST    LOAD    LOGIN   MAKE    MOUNT   PJOB    PLEASE  PPDT
PRESER  PRINT   PROTEC  PUNCH   QUEUE   R       REASSI  REENTE
RENAME  RESOUR  REWIND  RUN     SAVE    SCHEDU  SEND    SET
SKIP    SSAVE   START   SUBMIT  SYSTAT  TECO    TIME    TPUNCH
TTY     TYPE    UNLOAD  VERSIO  VI      YOADS   ZERO

THE MONITOR HAS THE FOLLOWING SET COMMANDS:
BLOCKS  CDP     CORMAX  CORMIN  CPU     CTEST   DATE    DAYTIM
DENSIT  DSKFUL  DSKPRI  HPQ     NOMESS  OPR     SCHEDU  SPOOL
TIME    TTY     WATCH
```

grees of sophistication needed for understanding spoken language for various tasks among which is the systems consultant. Their version of the systems consultant, called Voice-CC, requires a much more sophisticated system than ours because understanding spoken language is a more difficult and less understood task than understanding language written in machine-readable form. In one respect their task is easier because Voice-CC is to communicate with the user over a voice channel at the same time the user is trying to use the system over a conventional terminal. The system can know what the user has been doing, and this can be a great help in understanding what he is asking. We are proposing a consultant which operates via standard terminals. We will discuss a consultant that is independent of the system monitor, so that it has no auxiliary source of information on what the user might be attempting. (Though, if the consultant was part of the monitor, it could have this information.) Since the user is using the terminal to ask his questions, he is presumed to know such things as what the end of transmission character is; on the other hand, the sample protocol in Newell et al. [8, pp. 69–71] has a significant number of interactions on such topics. In either case, the task is much easier than a general natural language understanding system because the system's domain of competence is so limited, viz. the commands and features of the interactive system. We can assume that the user of the consultant wants information about these commands and that the request will be phrased in terms of the operations which these commands can perform. It is only necessary to recognize these terms and respond with a discussion of the relevant command and, possibly, related commands. The system need not understand the fine details of the user's request, just the gist of what he would like to do. Therefore, building the consultant is not much more difficult than writing a manual and providing a good index/thesaurus.

There is a controversy over whether natural language is an appropriate query language [5, 6, 7]. The opposing views seem to stress the ambiguities and general sloppiness of natural language. We trust that we have adequately explained that this is not an issue for the limited context we are discussing. There is another opposing view, however, that questions the usefulness of natural language input. This view is that habitual users of any system will prefer to use terse, formal language rather than natural language, which is generally verbose. The common response to this is that natural language input is best suited to "casual users." But do casual users exist? If so, who are they and what systems do (would) they use? Our answer is, "We have met the casual user, and he is us." Experienced programmers, when faced with a new system or with the need to use an unfamiliar feature on their old system, are casual users of the "help" program (system consultant). They use the system consultant because they do not know the command language, and they use it only until they learn the command language. What such a user wants is to be able to describe the operation he would like to perform and to be told the correct command to use in the given system. This is the natural language system consultant we are proposing.

Fig. 2(a). A session with the ELIZA Helper. User's input is on lines beginning with "/" or "?".

```
/-HELP
 A COPY OF THIS CONVERSATION IS BEING
 MADE FOR FURTHER STUDY...
 TO TERMINATE THIS PROGRAM, TYPE QUIT,
 THANKS, OR THANK YOU...
 HOW DO YOU DO.  WHAT IS YOUR PROBLEM?
? I NEED SOME INFORMATION ABOUT FFILES.
 FILES MAY BE ASSOCIATED WITH A JOB BY USING EITHER
 THE ATTACH  (DIRECT FILE)  OR  GET  (INDIRECT FILE)
 COMMANDS.  FILES MAY BE MADE PERMANENT BY USING EITHER
 THE DEFINE (DIRECT FILE) OR SAVE (INDIRECT FILE) COMMANDS
? I'M NOT SURE IF I WANT DIRECT OR INDIRECT FILES.
 A DIRECT ACCESS FILE IS A PERMANENT FILE WHICH, UPON
 REQUEST, BECOMES ASSOCIATED DIRECTLY WITH THE USERS
 JOB.  UNLIKE REQUESTS FOR INDIRECT ACCESS FILES, A
 COPY OF THE PERMANENT FILES IS NOT CREATED FOR USER
 ACCESS.  USE OF A DIRECT ACCESS FILE IS REQUESTED BY
 ENTERING THE ATTACH COMMAND, AND ALL SUBSEQUENT OPERATIONS
 ARE PERFORMED ON THE PERMANENT FILE ITSELF.  A DIRECT
 ACCESS FILE IS CREATED BY USE OF THE DEFINE COMMAND
? HOW IS THAT DIFFERENT FROM AN INDIRECT ACCESS FILE?
 AN INDIRECT ACCESS FILE IS A PERMANENT FILE THAT CANNOT
 BE ACCESSED DIRECTLY BY THE USER.  WHEN ACCESS IS REQUESTED
 BY A  GET  COMMAND, A COPY OF THE PERMANENT FILE IS
 CREATED AND ASSOCIATED WITH THE JOB MAKING THE REQUEST.
 NOTE THAT IF THE WORKING FILE IS ALTERED, THE USER
 MUST ENTER THE  REPLACE  COMMAND TO UPDATE THE PERMANENT
 FILE.  AN INDIRECT ACCESS FILE IS CREATED BY USE OF
 THE SAVE COMMAND
? WHAT ELSE CAN YOU TELL ME ABOUT INDIRECT ACCESS FILES?
 CAN YOU BE MORE SPECIFIC?
? SUPPOSE I WANT TO SAVE INDIRECT FILES.
 THE SAVE COMMAND CREATES AN INDIRECT ACCESS PERMANENT
 FILE.  THIS COMMAND ALLOWS A USER TO RETAIN A COPY
 OF THE SPECIFIED WORKING FILE IN THE PERMANENT FILE
/SYSTEM
? HOW DO I USE THE SAVE COMMAND?
 THE FORM OF THE SAVE COMMAND IS -
 SAVE,LFN=PFN/PW=PASSWRD,CT=N,M=M
? WHAT IS MEANT BY CT, I DON'T UNDERSTAND.
 CT REPPESENTS THE PERMANENT FILE CATEGORY.  THE FILE
 CATEGORY DETERMINES WHICH USERS MAY ACCESS A FILE.
 THERE ARE THREE CATEGORIES OF PERMANENT FILES - PRIVATE,
 SPRIV, AND PUBLIC
```

The ELIZA Helper

A natural language system consultant has been described briefly elsewhere [9]. To further demonstrate its feasibility, we have made Weizenbaum's ELIZA program [4, 10, 11] into a partial implementation.

The ELIZA program is actually quite simple and involves no sophisticated parsing, analysis or "understanding" of language. The input sentence is searched for predefined keywords; substitutions are made where specified; and the sentence is broken into phrases which can be used in the output sentence. Associated with each keyword is a level number which determines the preference of a response related to that keyword along with a list of reassembly rules to be used in the response. As the input sentence is scanned, a list of keywords in the sentence is constructed with the most preferred keyword at the front of the list. When the scan is completed, the decomposition pattern for the most preferred keyword is applied to the input sentence. If this pattern matches, parts of the input may be concatenated with a rule for that keyword to form the output sentence.

Fig. 2(b). A session with the ELIZA Helper.

```
? WHO MAY USE A FILE IF IT IS SAVED AS A PUBLIC FILE?
 PU REPRESENTS A PUBLIC FILE.  THESE FILES ARE AVAILABLE
 FOR ACCESS BY ALL USERS KNOWING THE FILE NAME, PASSWORD,
 AND USER NUMBER.  ONLY THE TOTAL NUMBER OF ACCESSES
 TO FILES OF THIS CATEGORY ARE RECORDED FOR THE ORIGINATOR
 OF THE FILE
? PLEASE TELL ME WHAT YOU MEAN BY M=M ON THE SAVE COMMAND.
 M REPRESENTS THE FILE OR USER PERMISSION MODE.  THERE
 ARE SEVERAL POSSIBLE MODES - WRITE, MODIFY, APPEND,
 READ, READMD, READAP, EXECUTE, OR NULL
? WHAT AM I ABLE TO DO WITH A FILE SAVED IN WRITE MODE?
 W REPRESENTS THE WRITE MODE AND ALLOWS THE USER TO
 WRITE, MODIFY, APPEND, READ, EXECUTE, OR PURGE THE
 FILE
? YOU MENTIONED WORKING FILES EARLIER AND I DIDN'T UNDERSTAND.
 A WORKING FILE IS EITHER A NEW FILE, CREATED BY THE
 USER, OR A COPY OF AN INDIRECT ACCESS FILE THAT ALREADY
 EXISTS IN THE SYSTEM.  ALL WORKING FILES ARE TEMPORARY
 IN NATURE AND CAN EXIST NO LONGER THAN THE USER IS
 LOGGED INTO THE SYSTEM.  A WORKING FILE IS A LOCAL
 FILE.  THE PARAMETER LFN ON MOST TIME-SHARING JOB COMMANDS
 AND PERMANENT FILE COMMANDS SIGNIFIES A LOCAL FILE
 NAME
? YESTERDAY I LOST A FILE BECAUSE I HADN'T USED IT IN AWHILE.
 DO YOU WANT TO KNOW ABOUT PRGLIST, WHICH GIVES A LISTING
 OF YOUR PURGED FILES, OR DO YOU WANT TO KNOW ABOUT
 PRGCOPY, WHICH WILL ALLOW YOU TO RETRIEVE A COPY OF
 A PURGED FILE?
? TELL ME ABOUT PRGCOPY.
 PRGCOPY IS A ROUTINE WHICH ALLOWS THE USER TO RETRIEVE
 A PERMANENT FILE WHICH HAS BEEN PURGED IN ACCORDANCE
 WITH THE PFILE PURGING ALGORITHM
? HOW DO I USE PRGCOPY?
 THE USER MAY ACCESS THE PRGCOPY ROUTINE BY THE COMMAND
 PRGCOPY,DATE,PFNAME.  WHERE DATE REPRESENTS THE NUMERIC
 DAY OF THE MONTH WHEN THE FILE WAS PURGED, AND PFNAME
 REPRESENTS THE NAME OF THE PFILE WHICH IS TO BE RECOVERED.
 THE DATE IS A REQUIRED PARAMETER AND IS AVAILABLE FROM
 PRGLIST.  IF PFNAME IS OMITTED, ALL OF THE USERS PFILES
 ARE RECOVERED FROM THAT DAYS PURGE TAPE.  NOTE THAT
 PFILES MAY NOT BE RECOVERED AFTER ONE MONTH.  ALSO
 NOTE THAT TO ACCOMPLISH THIS FEAT FROM THE TELETYPE,
 A SUBMIT COMMAND MUST BE USED
```

Fig. 2(c). A session with the ELIZA Helper.

```
? WHAT DOES THE SUBMIT COMMAND DO?
 THE SUBMIT COMMAND ALLOWS A VALIDATED USER TO CREATE
 A BATCH JOB DECK IMAGE AND SUBMIT IT TO THE LOCAL BATCH
 QUEUE AS A DEFERRED BATCH JOB.  THE USER CANNOT INTERACT
 WITH A JOB SUBMITTED FOR PROCESSING USING THIS COMMAND.
 RATHER, THE JOB IS TREATED IN THE SAME MANNER AS A
 JOB SUBMITTED FROM THE CARD READER AT WCC, BALANTINE,
 MEMORIAL, OR BUSINESS
? HOW DO I SUBMIT A JOB IN THIS WAY?
 THE FORM OF THE SUBMIT COMMAND IS -
 SUBMIT(LFN,Q,NR)C
? WHAT IS Q?
 Q REPRESENTS THE QUEUE SPECIFIED IN A SUBMIT COMMAND.
 IT MAY BE EITHER  B, N, OR E.  B MEANS THAT THE JOB
 OUTPUT IS DISPOSED TO THE BATCH QUEUE TO BE PRINTED,
 PUNCHED, ETC., AT WCC.  N MEANS THAT THE JOB OUTPUT
 IS DISPOSED TO THE BATCH QUEUE, BUT IS DROPPED AT JOB
 TERMINATION. (DEFAULT VALUE).  E MEANS THAT THE JOB
 OUTPUT IS DISPOSED TO EXPORT/IMPORT QUEUE FOR PRINTING
 AT A REMOTE SITE
? WHAT IS NR?
 NR REFERS TO THE NO REWIND OPTION.  IF PRESENT, THIS
 PARAMETER WILL INHIBIT REWIND AFTER EXECUTION OF ASSOCIATED
 COMMAND
? ONE MORE QUESTION - HOW DO I TURN THIS MACHINE OFF?
 TO LOGOFF FROM TELEX, TYPE GOODBYE OR BYE.  AN ALTERNATE
 WAY TO LOGOFF IS THROUGH THE HELLO OR LOGIN COMMANDS.
 THESE HAVE THE SAME EFFECT AS THE BYE COMMAND EXCEPT
 THE LOGIN SEQUENCE IS REINITIALIZED
? THANK YOU FOR YOUR HELP.
 MY PLEASURE
EXIT.
```

Where necessary, subsequent keywords from the input sentence are used. If no keywords are found or if all decomposition patterns fail to match, a stock sentence is chosen as the response.

The keywords, preference numbers, decomposition patterns, and reassembly rules are written on a file called the "script," which is read and used by the ELIZA program itself. By changing scripts, ELIZA can be made to perform different roles. ELIZA has been made to perform the system consultant role by choosing the keywords from the commands, uses of the commands and their synonyms. The reassembly rules are paraphrases or quotes from sections of the manual designed to provide information about the system and to answer the presumed questions.

The current implementation uses a SNØBØL 4 version of ELIZA [4][1] as a consultant for the KRONOS 2.1 operating system on the CDC 6600 computer at Indiana University. Specifically, the script is designed to provide help with permanent file (PFILE) commands used interactively, as well as knowledge about logging off the system, submitting files of control cards to the input queue, routing files to printers or card punches at various remote facilities, recovering an interactive job after being disconnected from the system, copying files, and other useful procedures as described in CDC's Time-Sharing User's Manual [2].

A user accesses the consultant by typing —HELP which activates a KRONOS procedure file, which then calls the ELIZA helper. It also copies every conversation for later study. Such sample conversations permit the maintenance of the script in a dynamic state to fit the specific needs of users and to provide navigational tools in the task of predicting conversation flow. A typical conversation is shown in Figure 2.

Organization of the consultant script follows the general theme that when a command name, a synonym of the command name, or a word implying some use of that command is recognized in a user input, the user is presumed to be asking for information about that command. The initial response is a general description of the usage of the command. It gives enough information that the user's question is probably answered, or at least the proper terminology is provided to rephrase the question.

If the same keyword reappears, the system responds with more specific information until the feature is completely described. The response to the next use of the keyword is:

CAN YOU BE MORE SPECIFIC? or
PLEASE DESCRIBE YOUR PROBLEM IN MORE DETAIL or
WHAT DO YOU MEAN BY _____? I DON'T UNDERSTAND

where _____ represents the input string. Further uses of the keyword are ignored, allowing less preferred keywords to determine the response.

Preference numbers determine dominance among keywords. Requests for information about parameters on control cards always dominate, since these keywords have a higher precedence than the simple name of a control card. If an input sentence is:

What does PW=PASSWORD mean on an ATTACH card?

the system responds relative to the keyword PW and describes what password should be specified when manipu-

[1] We actually use Cal SNOBOL, a smaller, faster version of SNOBOL4. The Cal SNOBOL version of ELIZA, as well as the ELIZA script and keywords, is contained in the appendices of [14].

lating a file rather than explaining more about the ATTACH command itself.

A more problematic situation occurs whenever the same keyword has differing interpretations, depending on the context. A partial solution is provided by assuming the user will remain within the overall context of a given script (an underlying assumption throughout ELIZA's history). Even with this assumption ambiguities arise. For example, the permanent file structure under KRONOS permits the specification of a mode under which a file may be accessed. These include a READ, WRITE, and APPEND mode. But in many situations, an input sentence may contain one of these keywords, though the user is not requesting mode information. An answer to this problem is provided in the ELIZA system through the use of more complex decomposition patterns. A phrase such as READ MODE may be specified as part of the pattern associated with the keyword READ so that responses relative to that word are not given indiscriminately. It is important to note that such disambiguation cannot always be accomplished in this manner. In some instances, ELIZA is made to respond with a question formulated to resolve the ambiguous keyword. For example, if an input sentence is:

How do I find the turnaround time at Marshal H. Wrubel Computing Center?

the system responds:

WOULD YOU LIKE TO DROP OR SUBMIT A JOB OR WOULD YOU LIKE TO SEE A STATUS OF THE QUEUES AT WCC?

Thus, a user is encouraged to use unambiguous keywords and is led to the eventual solution to his problem.

Summary

An excellent application for natural language understanding systems is an interactive system consultant. This is true for several reasons. The user of a system consultant is, ipso facto, not well versed in the system command language, and will cease using the consultant precisely when he does learn the command languguage. He is, therefore, precisely the kind of user best served by a natural language input system. On the other hand, the system consultant operates on a very restricted domain, viz. the system commands and the uses to which they may be put. At this time, natural language understanding systems have been successful when applied to restricted domains, and they have been successful only in such applications. Furthermore, the system consultant does not require a fine understanding of the input. It is acceptable if the consultant merely recognizes what command or feature is being inquired about and launches into a discussion of that feature. To demonstrate the feasibility of a natural language system consultant, we have implemented one using ELIZA, a keyword oriented conversation program.

Received September 1974; revised October 1974

References

1. Codd, E.F. Seven steps to rendezvous with the casual user. Proc. IFIP TC-2 Working Conf. on Data Base Mgmt Syst., Cargese, Corsica, North-Holland, Amsterdam, April 1–5, 1974.
2. Control Data Corporation. Time-sharing user's reference manual, Pub. No. 60407600, Tech. Pub. Dep., Arden Hills, Minn., July 1973.
3. DEC System-10 Users Handbook, Digital Equipment Corporation, Maynard, Mass., 1972.
4. Duquet, R.T. ELIZA in SNOBOL4. SIGPLAN Notices 5, 12 (Dec. 1970), 50–60.
5. Guiliano, Vincent E. In defense of natural language. Proc. ACM Ann. Conf., ACM, New York, 1972.
6. Hill, I.D. Wouldn't it be nice if we could write computer programs in ordinary English—or would it? *The Computer Bulletin 16*, 6 (June 1972), 306–12.
7. Montgomery, Christine A. Is natural language an unnatural query language? Proc. ACM Ann. Conf., ACM, New York, 1972.
8. Newell, A., Barnett, J., Forgie, J.W., Green, C., Klatt, D., Licklider, J.C.R., Munson, J., Reddy, D.R., and Woods, W.A. *Speech Understanding Systems—Final Report of a Study Group.* American Elsevier, New York, 1973.
9. Pirtle, Melvin W. Help. In *Conversational Computers*, William D. Orr (Ed.), John Wiley, New York, 1968.
10. Weizenbaum, J. ELIZA—a computer program for the study of natural language communications between man and machine. *Comm. ACM 9*, 1 (Jan. 1966), 36–45.
11. Weizenbaum, J. Contextual understanding by computers. *Comm. ACM 10*, 8 (Aug. 1967), 474–80.
12. Winograd, Terry. *Understanding Natural Language*, Academic Press, New York, 1972.
13. Woods, W.A., and Kaplan, R.M. The lunar sciences natural language system. *BBN Rep. No. 2265*, Bolt Beranek and Newman, Cambridge, Mass., Sept. 1971.
14. Shapiro, S.C., and Kwasny, S.C. Interactive consulting via natural language. Tech. Rep. No. 12, Comput. Sci. Dep., Indiana U., Bloomington, Ind., June 1974.

Corrigendum
Programming Techniques

Jean C. Vaucher and Pierre Duval, A Comparison of Simulation Event List Algorithms. *Comm. ACM 18*, 4 (Apr. 1975), 223–230.

P.229: The graph under heading "(b) Post-order tree" belongs under "(d) Indexed list"; and the graph under (d) belongs under (b).

Corrigendum
Numerical Mathematics

J. Todd, The Lemniscate Constants, *Comm. ACM 18*, 1 (Jan. 1975), 14–19.

P.16, 2d column, line −8: for (1/2) read (1/4).

P.17, 3d line after Theorem 11 should read: $\frac{1}{2}\pi = 3 \arcsin \frac{1}{2}$.

P.18: Theorem 15 should read:
$\lim M(1, x) \log x^{-1} = \frac{1}{2}\pi$, as $x \to 0$.

P.18: 3d line after formula 6.1 should read:
$\vartheta_2(q) = 2q^{\frac{1}{4}}[1 + q^2 + \cdots]$.

P.19, add reference:
51. Fuchs, W. Das arithmetisch-geometrische Mittel in den Untersuchen von Carl Friedrich Gauss. *Gauss-Gesellschaft Göttingen Mitt. 9* (1972), 14–38.