Spelling Correction
for an Intelligent Tutoring System

Yoon Hee Lee  Martha Evens
Illinois Institute of Technology
Chicago, Illinois 60616

Joel A. Michael  Allen A. Rovick
Rush Medical College
Chicago, Illinois 60612

Abstract

Our spelling correction program is part of a system for understanding ill-formed input in an intelligent tutoring system for medical students. Speed and user-friendliness were the most important considerations in the design. The system can correct most kinds of spelling errors including order reversal, missing characters, added characters, and character substitutions. It also handles novel abbreviations and word boundary errors. It is implemented on a Xerox 1108 AI machine in Interlisp-D. The lexicon is stored in a trie structure to speed up searching. We provide the students with a full-screen editor and an input tracer to simplify the input process as much as possible.

1 Introduction

This paper is concerned with the development of the spelling correction component of a program that tries to deal with ill-formed student input in a tutoring context. This implementation of our ideas was written in Interlisp-D on a Xerox 1108 Artificial Intelligence machine. This research forms part of a larger project – the design of a computer-aided instruction program to assist first-year medical students in solving problems in cardiovascular physiology. We are starting with an existing CAI program CIRCSIM [1] and attempting to build a succession of increasingly intelligent versions. The goals of our research project are add facilities to understand ill-formed student input, model student knowledge, and generate natural language explanations tailored to individual students [2].

1This work was supported by the Cognitive Science Program, Office of Naval Research under Grant No. N00014-89-J-1952, Grant Authority Identification Number NR4422554 to Illinois Institute of Technology. The content does not reflect the position or policy of the government and no official endorsement should be inferred.
The tutor provides users of the system with a full screen text editor for use in text input. This has advantages both for us and for the student. It makes the student comfortable and it makes it easy for the student to correct his/her own mistakes – leaving fewer problems for us to deal with. The programmable Interlisp-D text editor, TEdit, proved to be a very satisfactory tool for this purpose, making it easy to update and trace the student input. A trace facility is essential to a program supporting interactive input. We use a temporary buffer for input tracing. As a result, students can return to the latest updated version of their text input and make changes at any point.

2 Sublanguage

A sublanguage is a form of natural language used by a particular community of specialists to describe a particular subject matter [3,4]. In order to analyze natural language reliably, a computer system requires a great deal of information about the syntax of the language, about the structure of the discourse, and about the subject matter for the language. Because of the need for detailed knowledge of the subject matter, natural language input systems perform better within a very limited domain of discourse. For this project we needed to analyze the student cardiovascular sublanguage and formulate a sublanguage grammar and lexicon.

We collected and studied student language samples collected in face-to-face tutoring sessions. We are now collecting still more examples from keyboard-to-keyboard tutoring sessions. We made KWIC indexes and concordances of the student language samples using the VAX computer system. Then we used a phrase counting program to isolate common phrases and added them to the lexicon. In the process of sublanguage analysis, we found about 1650 words and phrases belonging to the student cardiovascular sublanguage. We are using a Lexical Functional Grammar parser built by Lee to discover sublanguage specific grammar rules as well.

3 Lexicon

The lexicon contains essential data entries for the spelling correction system and the parser/understander. Each lexical entry contains orthographic, syntactic, and semantic information so that the system can retrieve information for both spelling correction and the parser/understander at the same time.

The lexicon data structure is very important for the spelling correction and parsing system because the search algorithm depends on this structure. These, in turn, determine program run time. We decided to use the kind of tree structure call a trie [5]. The trie is an index structure that is particularly useful when key values are of varying size. A trie is a tree of degree m > 1 in which the branching at any level is determined not by the entire key value but only a portion of it.