Chapter 24

Querying a Static and Dynamic Learner Corpus

Trude Heift
Simon Fraser University, Canada

Catherine Caws
University of Victoria, Canada

ABSTRACT

This chapter discusses the cyclical process of collecting and recycling learner data within the E-Tutor CALL system and presents a study on student usage of its data-driven learning (DDL) tool. E-Tutor consists of a static and dynamic learner corpus for L2 learners of German. The static learner corpus has been constructed from approximately 5000 learners who used the system over a period of five years. These learners provided millions of submissions from a variety of activity types. In addition, all concurrent E-Tutor users contribute data to a dynamic corpus, which allows them to compare and examine their ongoing system submissions to those contained in the static corpus. The authors conducted a study with 84 learners and recorded their interaction with the DDL tool of E-Tutor over one semester. Study results on student usage suggest that investigating sample input of a large, unknown user group might be less informative and of less interest to language learners than their own data. For the DDL tool to be useful for all proficiency levels, training and scaffolding must also be provided.

INTRODUCTION

Data collections from learner-computer interactions are commonly used to determine learning outcomes and to assess and improve learning tools and software features that are designed to enhance the user’s learning experience. By focusing on the learners’ needs, a holistic and cyclical approach to software engineering is generally preferred because each and every stage during the lifecycle delivers output that serves as input for the subsequent stage with evaluation being an essential component after each step (see Caws, 2013; Hamel, 2012; Hamel & Caws, 2010; Hémard, 1999; Hubbard, 2011; Levy, 2002). A much less explored application of learner data collections and analyses, however, uses recycled learner data to expand an existing CALL environment with data-driven learning (DDL), whereby the learner...
corpus used for DDL consists of authentic learner submissions collected from the very same learning environment (see Caws & Hamel, 2013; Heift & Caws, 2014).

DDL has been explored under various aspects. It generally emphasizes the ways in which it can facilitate the implementation of a methodology for language learning that is centred on authenticity in contents, contexts, and tasks (van Lier, 1996), commonly achieved with a corpus of L1 or L2 data. This article focuses on the recycled data of E-Tutor, a comprehensive Intelligent Computer-assisted Language Learning (ICALL) environment for L2 German. We discuss the multiple DDL applications of the collected user data of E-Tutor, mainly with respect to the learner but also the instructor and the researcher. E-Tutor contains a static learner corpus which was constructed from user data collected from approximately 5000 students who completed a variety of activity types over a period of five years. In addition, the system constructs a dynamic learner corpus during system usage in runtime from submissions of concurrent learners. E-Tutor follows a cyclical process of development, implementation, and evaluation to inform language teaching pedagogy and provide system enhancements generated by the outcomes of data collections, in particular, with regard to interface design, error analysis, help options, system feedback, and DDL.

In the following, we first provide a theoretical perspective on the cyclical process of language software engineering by introducing the ADDIE model and then illustrate its applicability to DDL in the context of a learner corpus collected in the same CALL environment in which it is being used. We briefly introduce E-Tutor by describing its system functionality and the learner corpus that we constructed from its recycled learner data. Next, we discuss the learner corpus with respect to its applications and uses for the learner, instructor and researcher. The article concludes with a discussion of opportunities for further system extensions and research.

A CYCLICAL APPROACH TO LANGUAGE SOFTWARE ENGINEERING

A traditional, commonly employed conceptual framework for instructional system design is the ADDIE model (analysis, design, development, implementation, evaluation), which represents a guideline for building effective training and performance support tools (see Colpaert, 2004; Forest, 2014; Molenda, 2003; Mutlu, 2016). As explained by Colpaert (2006a), this framework presents the advantage of being compatible with a “research-based” and “research-oriented” (RBRO) methodology (p. 115). The initial ADDIE model is said to originate from a former instructional design (ID) method based on a five-stage sequential design process where progress occurs within a hierarchical, linear structure (Forest, 2014; Strickland, 2006). The basic idea of such a hierarchical model is that a phase of a learning program or software product’s lifecycle is finished perfectly before moving to the next phases, and that the outcome of one phase informs the next phase so that decisions can be made in a logical and more efficient manner. A common and obvious criticism towards such a hierarchical model is that one may not always know exactly what requirements are needed before reviewing a working prototype and reacting to it. Thus practitioners have to frequently revise the computational as well as pedagogical steps making this process inefficient and costly. As a result of these objections, the original hierarchical version of the ADDIE model eventually became more dynamic and interactive and, by the mid-1980s, the version to which we are familiar today appeared (Forest, 2014).

As illustrated in Figure 1, ADDIE is an acronym that stands for the five main phases of the model: Analysis (define needs and constraints), Design (specify learning activities, choose methods, etc.), De-