Constructing a Data-Driven Learning Tool with Recycled Learner Data

Trude Heift, Department of Linguistics, Simon Fraser University, Burnaby, Canada
Catherine Caws, French Department, University of Victoria, Victoria, Canada

ABSTRACT

This paper discusses a data-driven learning (DDL) tool, which consists of a learner corpus for L2 learners of German. The learner corpus, in addition to submissions from ongoing current users, has been constructed from millions of submissions from a variety of activity types of approximately 5000 learners who used the E-Tutor CALL system over a period of five years. By following a cyclical process of development, implementation, and evaluation, adapted from the ADDIE model, E-Tutor helped us not only to inform language teaching pedagogy and to provide system enhancements generated by the outcomes of vast data collections, but also to expand an existing learning environment (e.g., Tutorial CALL) to include DDL. The article discusses the cyclical process of collecting and recycling learner data by also focusing on the design features of the DDL tool of E-Tutor within the ADDIE framework and providing data on student usage.

Keywords: ADDIE, CALL Design, Corpora, DDL, German as a Foreign Language

1. 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 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; Hubbard, 2011; Hamel & Caws, 2010; Hémard, 1999; 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), especially whereby the learner corpus used for DDL consists of authentic learner submissions from the very same learning environment.

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. While previous studies using E-Tutor discussed learner feedback (Heift, 2004), error correction (Heift & Rimrott, 2008), and learner variability (Heift, 2008), among others, the present paper focuses on 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 learner corpus that was mainly constructed from user data collected from approximately 5000 students who completed a variety of activity types over five years. Accordingly, E-Tutor follows a cyclical process of development, implementation, and evaluation to inform language teaching pedagogy and to 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 as that in which it is being used. We 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.

2. A CYCLICAL APPROACH TO LANGUAGE SOFTWARE ENGINEERING

A traditional, commonly employed model for instructional system design is the ADDIE framework (analysis, design, development, implementation, evaluation), which represents a guideline for building effective training and performance support tools (see Colpaert, 2004). As explained by Colpaert (2006a), this model presents the advantage of being compatible with a “research-based” and “research-oriented” (RBRO) methodology (p. 115).

The initial ADDIE model was based on a sequential design process, or the so-called waterfall model, in which progress is seen as flowing steadily downwards, like a waterfall, through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Production/Implementation, and Maintenance (see Strickland, 2006). The basic idea of such a waterfall model is that a phase of a software product’s lifecycle is finished perfectly before moving to the next phases and learning from them. 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 commenting on it and thus practitioners have to frequently revise the steps which is 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 familiar today appeared (see Strickland, 2006).

As illustrated in Figure 1, and mentioned above, 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.), Development (begin production), Implementation (put the plan into action), and Evaluation (evaluate the plan from all levels for next implementation). In ADDIE, each step has an outcome that feeds information into the subsequent step and in order to achieve a better “design fit” (Levy, 1997, p. 163), ongoing evaluation after each step is essential. Indeed, according to Colpaert (2004, 2006b), this iterative, circular, nonhierarchical, and transformational process of software development has also become the most widely accepted view on CALL courseware design (see also Shneiderman, 1997). From a pedagogical
Related Content

Going beyond Audio: Adding Multimedia to Podcasts for Foreign Language Education
www.igi-global.com/chapter/going-beyond-audio/50669?camid=4v1a

The Benefits of Student-Led Video Production in the Language for Business Classroom
Claudia Gremler and Elisabeth Wielander (2018). Teaching Cases Collection (pp. 155-193).
www.igi-global.com/chapter/the-benefits-of-student-led-video-production-in-the-language-for-business-classroom/186530?camid=4v1a
Planning for Future Inquiry: Gaps in the CALL Research
www.igi-global.com/article/planning-for-future-inquiry/209398?camid=4v1a

Constructing a Collocation Learning System from the Wikipedia Corpus
www.igi-global.com/article/constructing-a-collocation-learning-system-from-the-wikipedia-corpus/164912?camid=4v1a