Tactical Language and Culture Training Systems:
Using Artificial Intelligence to Teach Foreign Languages and Cultures

W. Lewis Johnson and Andre Valente

Alelo, Inc.
11965 Venice Bl., Suite 402
Los Angeles, CA 90066 USA
ljohnson@alelo.com; avalente@alelo.com

Abstract

The Tactical Language and Culture Training System (TLCTS) helps people quickly acquire communicative skills in foreign languages and cultures. More than 20,000 learners worldwide have used TLCTS courses. TLCTS utilizes artificial intelligence technologies in multiple ways: during the authoring process, and at run time to process learner speech, interpret learner actions, control the response of non-player characters, and evaluate and assess learner performance and proficiency. This paper describes the architecture of TLCTS and the artificial intelligence technologies that it employs, and presents results from multiple evaluation studies that demonstrate the benefits of learning foreign language and culture using this approach.

Introduction

The Tactical Language and Culture Training System (TLCTS) helps people quickly acquire functional skills in foreign languages and cultures. It includes interactive lessons that focus on particular communicative skills, and interactive games that apply those skills. Heavy emphasis is placed on spoken communication: learners must learn to speak the foreign language to complete the lessons and play the games. It focuses on the language and cultural skills needed to accomplish particular types of tasks, and gives learners rich, realistic opportunities to practice achieving those tasks.

Several TLCTS courses have been developed so far. Tactical Iraqi[TM], Tactical Pashto[TM], and Tactical French[TM] are in widespread use by US marines and soldiers, and increasingly by military service members in other countries. Additional courses are being developed for use by businessmen, workers for nongovernmental organizations, and high school and college students. While precise numbers are impossible to obtain (we do not control copies made by the US Government), over 20,000 and as many as 50,000 people have trained so far with TLCTS courses. Over 1,000 people download copies of TLCTS courses each month,

either for their own use or to set up computer language labs and redistribute copies to students. Just one training site, the military advisor training center at Ft. Riley, KS, trains approximately 10,000 people annually.

Artificial intelligence technologies play multiple essential functions in the TLCTS courses. Speech is the primary input modality, so automated speech recognition tailored to foreign language learners is essential. TLCTS courses are populated with "virtual humans" that engage in dialog with learners. AI techniques are used to model the decision process of the virtual humans and to support the generation of their behavior. This makes it possible to give learners extensive conversational practice. Learner modeling software continually monitors each learner's application of communication skills to estimate the learner's level of mastery of these skills. This helps instructors and training supervisors to monitor learners' progress, and enables the software to guide learners to where they need to focus their training effort. Artificial intelligence is also integrated into the systems' content authoring tools, assisting content authors in the creation and validation of instructional content.

System Overview

TLCTS courses are currently delivered on videogame-capable personal computers, equipped with headset microphones. Each course contains a set of interactive Skill Builder lessons, focusing on particular communicative skills. Figure 1 illustrates an exercise page from the Skill Builder in the Tactical French course on the language and culture of French-speaking sub-Saharan Africa. The figure is an example of an utterance formation exercise where the learner responds to a written prompt by speaking an utterance in the target language. Here the learner must think of an appropriate way to say goodbye. The learner said "Salut". As the system feedback indicates, this is an inappropriate way to address a Chadian woman that the learner has just met. Depending on the type of exercise, the system can give feedback on pronunciation, morphological and grammatical forms, word choice, or cultural pragmatics, as in this example.

Games play an essential role in TLCTS courses, providing essential opportunities. Each course incorporates a scenario-based Mission Game, where learners play a character in a 3D virtual world that simulates the target culture. Figure 2 shows a screenshot from the Mission Game in the Canadian Forces version of Tactical Pashto.

Figure 1. Example Skill Builder exercise

Figure 2. Example Mission Game dialog

In the Mission Game, the learner's avatar must carry out a mission that involves interaction with the local people. In Figure 2, the player's character (center, crouching) has just entered an Afghan village and initiated a conversation with a group of children. The boy in the foreground under the vertical arrow is the current focus of conversation; however, the other children nearby are also listening in and may jump into the conversation as appropriate. The face icons on the right margin are Rapport Meters, which indicate the current attitudes of the children toward the player.

The Alelo Architecture

The architecture underlying TLCTS (Figure 3) supports a set of complementary training products built from a com-

mon set of content specifications. All content in TLCTS courses is specified in XML and stored in a Web-compatible content repository. A Web portal named *Kona*[1] provides access to the repository and supports collaborative editing. Each project is represented in *Kona* as a "book", containing a series of "chapters" (each specifying an individual lesson or game scene). Project members can check out individual chapters (i.e., lock them for editing), update them using Web-based authoring tools, and then check them back in. An underlying version-control framework (Subversion) prevents simultaneous edits and allows roll-back to earlier versions if necessary. Our goals have been to make the authoring cheaper, faster, better, and easier, and to remove the need for engineers and researchers to be involved in authoring and production.
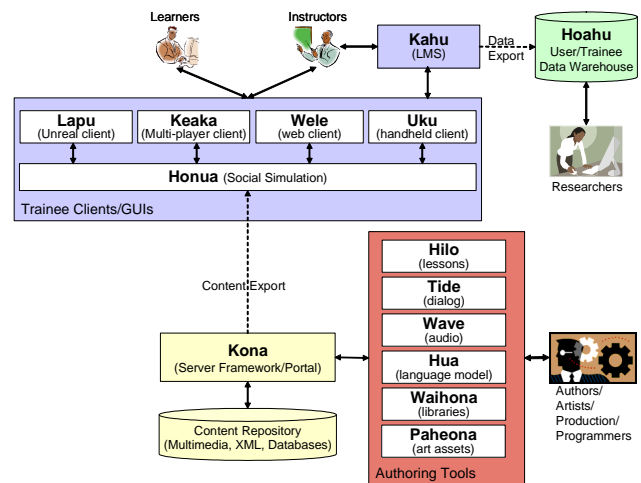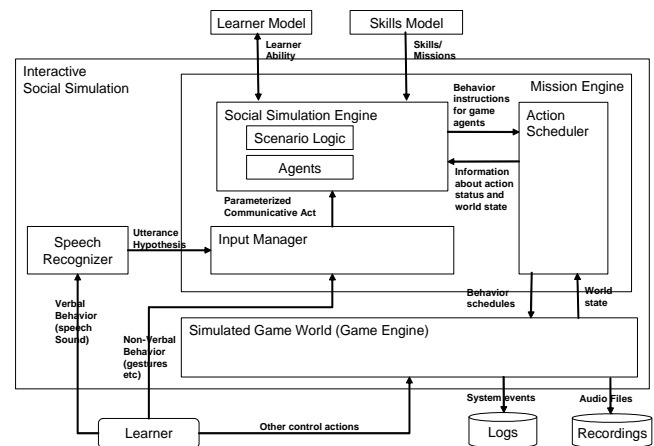

Figure 3. Overall Alelo architecture

Figure 4. Lapu client simulation architecture

The architecture supports several delivery platforms. Our *Lapu* client runs on a Windows PC equipped with a headset microphone. *Lapu* is built on top of Epic Game's Un-

[1] Many of the architecture components have Hawaiian names, reflecting the first author's connection with Hawaii (see table at the end of the article for a glossary.).

real Engine 2.5, which handles scene rendering and provides a set of user interface classes. Figure 4 shows the *Lapu* architecture in further detail, focusing on the support for social simulations (dialogs with animated characters). The learner communicates with characters in the game, using voice and gestures selectable from a menu. The Input Manager interprets the speech and gestural input as a communicative act (i.e., speech acts augmented with gestures). The Social Simulation Engine determines how the environment and the characters in it respond to the learner's actions. The character actions are realized by the Action Scheduler, which issues animation commands to the Unreal engine. As the learner interacts with the system, the Learner Model is updated to provide a current estimate of learner skill mastery. Interaction logs and learner speech recordings are saved for subsequent analysis. These components are implemented primarily in Python, with some supporting routines in C++ and UnrealScript (the scripting language for the Unreal Engine). Reliance on UnrealScript is minimized to facilitate porting to other game engines.

Handheld device implementations (*Uku*) allow trainees to continue their training when they do not have access to a PC. One version of *Uku* currently under beta test delivers TLCTS Skill Builder content on iPods. Media assets (instructional texts, recordings, images, and videos) are extracted from the repository, converted into iPod-compatible formats, and organized using the iPod Notes feature[2]. Example dialogs between animated characters are converted into videos. We developed a prototype for the Sony PlayStation Portable, and we are evaluating other handheld platforms. All clients except for the iPod provide interactive exercises. In cases where we are able to port the speech recognizer to the handheld device, the client implementation utilizes the speech recognizer; for platforms that cannot run the speech recognizer effectively, speech recognition input is replaced with alternative methods, such as selection from menus of possible utterances.

A new Web-based client named *Wele* is also being increasingly used to deliver content. *Wele* currently supports just the Skill Builder; however, we plan to extend it to include Mission Game implementations. *Wele* is accessed via a Web browser, typically Internet Explorer, and is implemented in the Adobe Flex[3] Internet application framework. *Wele* runs the speech recognizer on the client computer as a plug-in. Interactive dialogs and animations are realized as Flash files inserted into a Web page.

A lightweight learning management system called *Kahu* communicates with the client computers over a local network. Our users frequently organize their training computers in learning labs using ad hoc local networks (using Windows shares), then move their computers and reorganize them in different configurations. *Kahu* provides and manages repositories for learner profiles and supports easy reconfiguration and disconnected operation of these labs. The system provides mechanisms for a training manager to create and configure users and groups of users, and produce training reports. We also use it to help retrieve learner profiles, system logs, and recordings from training sites and store them in a data warehouse, where they may be retrieved for further analysis. The recordings are used to retrain and improve the speech recognizer, while the logs and learner profiles are used in research to understand how the users employ the training system, and to suggest areas for further improvement of the software.

Application Development and Deployment

TLCTS system developed over several years, starting as a research project at the University of Southern California's Information Sciences Institute and later at our company. The original project at USC spanned about four years starting in 2003, and its continuation at Alelo started in 2005. Costs for the development of the underlying science and technology are around $5M; costs for the engineering work to make it deployable were around $1M, and course development costs have gone down from the $800K range to $300K-$600K, depending upon the target language and culture. This was first funded by DARPA and U.S. Special Operations Command (USSOCOM), and more recently with by US Marine Corps, Army, and Air Force.

Throughout the project we adopted an iterative development strategy. The goal is to produce new versions of the system every several months, deploy and evaluate these new versions, and use the evaluation results to inform the next spiral. We believe that research and development benefit from use and practice, and that each real-life testing stage provides critical feedback. Indeed, we believe that a good deal of the success of TLCTS stemmed from acting upon the feedback from our users.

Some of the key development challenges were related to delivery. The transition from a research prototype to a widely used application required significant engineering effort to develop good installers, integrate the components more closely, improve performance, add user interface polish, etc. Early versions of the architecture were intended as a platform for experimentation (e.g., using agent-oriented integration mechanisms), a concept that was useful for research purposes but not stable or fast enough for a deployed system. At several points, we had to sacrifice flexibility for performance, stability, and ease of use. For example, the original implementation of the Skill Builder was build on top of a courseware platform (ToolBook), but users found it cumbersome to switch back and forth between separate Skill Builder and Mission Game applications. We therefore converted the Skill Builder to run on top of the game engine, which was unfortunately less flexible – for example, only later we were able to recover the ability to play videos in our lessons.

---

[2] http://developer.apple.com/ipod.

[3] http://www.adobe.com/products/flex/

As with all large applications, our software has gone through several re-architecture steps. Nonetheless, we have extended the system without any major rewrites. Maintenance also means authoring new training content and improving existing content. This requires the authoring tools discussed in Section "Authoring" and maintenance of the representations. Our use of XML has allowed us to evolve our content across system versions.

We made several attempts to create an authoring suite, some more successful than others. Our content development methodology has evolved significantly over time, and our authoring tools have adapted as well. We have come to recognize the collaborative nature of content development. The original implementation of an authoring tool was based on a concept of a standalone desktop tool used by one author at a time – an unrealistic assumption. The latest iterations have embraced the idea of a web application, and emphasized features to manage the collaboration process and cater to the specific needs of the different types of users (linguists, artists, production staff, programmers, etc.)

Uses of AI Technology

Artificial intelligence technologies are employed extensively in the Alelo run-time environments and authoring tools. The following is a summary of the roles that artificial intelligence currently plays, starting with the run-time environment. However, in practice, authoring concerns and run-time concerns are inextricably linked; run-time processing methods cannot be employed if they place an unacceptable burden on authors to create content for them.

Speech Recognition

TLCTS is particularly ambitious in its reliance on speech recognition technology. Recognition of learner speech is particularly demanding and challenging. Beginning language learners sometimes mispronounce the foreign language very badly, and blame the software if it is unable to recognize their speech. The speech recognizer also needs to be very discriminating at times, and accurately detect speech errors to provide feedback to learners. Typical speech recognizers, in contrast, are designed to disregard speech errors and focus on decoding speech into text. Our approach must apply equally to common languages such as French and to less commonly taught languages such as Pashto or Cherokee, for which few off-the-shelf speech recognition resources exist. For these reasons we rejected prepackaged speech recognition solutions and opted to develop our own tailored speech recognition models.

We employ hidden Markov acoustic models developed using the Hidden Markov Model Toolkit (HTK)[4] and delivered using the Julius[5] open source speech recognition toolkit. Models are developed iteratively in a bootstrapping

---

[4] http://htk.eng.cam.ac.uk

[5] http://julius.sourceforge.jp/en_index.php

process. We construct an initial acoustic model from a combination of licensed corpora, an initial corpus of learner speech, and sometimes corpus data from similar languages, and integrate it into the first beta version of the TLCTS course. We then recruit beta testers to start training with the software. It records samples of their speech, which we use to retrain the speech recognizer. Each successive version of a TLCTS speech recognizer has improved performance over the previous versions because it is trained on speech data collected from the previous versions.

The speech recordings must be annotated to indicate the phoneme boundaries. Speech annotation is time-consuming but has a critical effect on the quality of the resulting acoustic model. Corpora licensed from other sources often have annotation errors and need to be re-annotated.

Each language model includes a grammar-based model built from phrases extracted from the authored content, and a default "garbage model" constructed with a fixed number of acoustic classes, states, and probability mixtures. If the learner speaks an utterance that is in the recognition grammar, the grammar-based model will usually recognize it with higher confidence than the garbage model does, and if the utterance is out of grammar the garbage model will recognize it with higher confidence. The garbage model rejects many utterances that the learners know are incorrect, and forces learners to speak the language with at least a minimum level of accuracy. By adjusting the properties of the garbage model we can adjust the recognizer's tolerance of mispronounced speech.

Language learners normally produce several types of errors in their speech, including pragmatic errors (e.g., failure to adhere to cultural norms in discourse), semantic errors (e.g., word confusions), syntactic errors, morphological errors, and pronunciation errors. Where possible we use the speech recognizer to explicitly detect these errors, using language models that are designed to detect language errors. Pronunciation error detection is handled as a special case. Earlier versions of TLCTS attempted to detect pronunciation errors on a continual basis in the Skill Builder (Mote et al., 2004). However, evaluations identified problems with this approach: it was difficult to detect pronunciation errors reliably in continuous speech (leading to user frustration), and the continual feedback tended to cause learners to focus on pronunciation to the exclusion of other language skills. We have since adopted a different approach where the system does not report specific pronunciation errors in most situations, but instead provides a number of focused exercises in which learners practice particular speech sounds they have difficulty with.

Speech recognition performance depends in part on the authored content, and so we have developed authoring guidelines and tools that enable content authors to create learning materials suitable for speech recognition, without requiring detailed knowledge of the workings of speech

recognition technology. When authors write dialogs and exercises, they specify alternative ways of phrasing these utterances, some of which are well-formed and some of which may illustrate common learner errors. These are used to create recognition grammars that recognize learner speech as accurately as possible. Authors can mark phrases as "ASR-hard", meaning that the Automated Speech Recognizer (ASR) should not be applied to them and they should be excluded from the speech recognition grammar. Since speech recognition accuracy tends to be better on longer phrases, individual words (which sometimes appear on lesson pages) are sometimes excluded in this way.

We have used learner data to evaluate speech recognition performance for the current languages (Iraqi Arabic, French, Pashto) and obtained recognition rates of 95% or above for in-grammar utterances. However the training data is skewed towards male users and beginner users (reflecting our current user population), which makes the performance degrade for female and advanced users.

Despite these positive results, users have raised issues about the speech recognition function. For example, some users complained that the recognizer did not recognize what they thought were valid utterances. These problems can be caused by several reasons. First, some acoustic models (e.g., for French) were built using proportionally more native data (than non-native) and as such are less forgiving of mispronounciations. Other users complain that they found the Automated Speech Recognizer (ASR) too lenient. We have responded to these problems in part by allowing the learner to individually adjust the garbage model (see above) to make it appear more or less lenient of mistakes. A second reason for these problems is that the ASR uses grammars that change for each specific part of the system. These grammars are compiled from existing content, and thus reflect whatever subset of the language we are teaching. Therefore, a native or fluent speaker usually manages to produce an utterance that is not recognized because it is outside of the system's grammar. We have been working on techniques to extend these grammars while maintaining recognition rates (Meron, Valente and Johnson, 2007). Finally, many complaints assumed unrealistic expectations as to the type of noise the system could sustain while keeping recognition rates. A user sent us some system recordings with a loud TV program on the background, and still complained the recognizer was not working well. In response, we developed a speech quality assessment module that gives the learner feedback about possible indications of poor voice recording—levels too low or too high, excessive periods of silence, noise, or indications that the beginning or end of the recording was clipped. This is displayed first as the background color on a small waveform display — red for poor recording quality, green for good, yellow for marginal. If a user clicks on that display, a larger window appears that compares the learner's speech with the native speaker's.

Modeling Dialog

One key challenge for the dialogs in TLCTS is to manage the variability of the spoken inputs a learner may produce. On one hand, we wish to attain high variability, meaning that the system should recognize, properly interpret, and react to a large subset of the language. On the other hand, we wish to author interactions that train specific language and culture skills,not unconstrained dialog. Furthermore, increasing the size of the language subset makes speech recognition and interpretation much harder. Our strategy to balance those needs is to manage variability in two layers. In the speech layer, we constrain the speech to be recognized in such a way that we can limit the possible utterances and improve ASR performance as described above. In the act layer, we manage the possible acts the user can perform and the mapping from speech to acts.

We originally created linear scripts and branched scripts, and annotated each utterance with an act name. We then manually translated the text scripts into hard-coded programs. This translation included the logic for plot progression and character control, as well as the mapping from user input (the text output by the ASR) to actions in the game world (in the form of a list of input-action pairs). However, we found out that scripts were too restrictive a medium in that they limited the act variability. We could improve the situation by manually programming more complex algorithms in software code, but that approach it required a programmer to complete the scene (which was both expensive and non-scalable), and it was impossible to verify if the code was consistent with the author's intent.

In our current methods, authors write specifications of dialog interactions at the act level. To increase the variability of dialog that the framework supports, we introduced utterance templates into the dialog specifications. An utterance template is a combination of a more flexible grammar definition syntax (which allow increased variability in user inputs), with additional syntactic constructs used to specify semantic content allowing parameterized utterances (which increases variability in possible responses). The grammar is passed directly to the speech recognizer, and when an utterance is parsed it comes with semantic annotation that indicates information such as the type of action or action-object specified in the utterance. We have developed tools maintaining these utterance templates, and have been working on mechanisms for managing libraries of utterance templates (Meron, Valente and Johnson, 2007).

Believable Virtual Humans

A central feature of our approach is to put the user in a social simulation with non-player characters–*virtual humans*. Our original virtual human behavior generation pipeline was relatively simple. An input from the player would be provided as an abstract act (say, *greet_respectfully*) to an agent implemented in PsychSim (Si et al., 2005). The agent system would specify an act for its NPC to perform (e.g., *inquire_identity*). An XML

file would specify the utterance and any animations that needed to be performed to realize that act; sometimes also a small script in Python would be called to perform more complex physical actions (say, walk over a specific place). More recently, we have adopted a variant of the SAIBA framework (Vilhjalmsson and Marsella, 2005), that separates intent planning (the choice of courses of action – to complain, cooperate, make a request, etc. – that are adequate to the situation at hand) from the production of believable physical behavior.

We originally used the PsychSim agent system to perform intent planning (Si et al., 2005). While PsychSim is highly expressive, its agents are extremely hard to author and computationally complex. The modeling language is not well suited to represent dialogs or world knowledge. We then turned to a finite-state machine approach which authors can use to specify moderately complex scenarios, and has proved scalable to large numbers of dialogs (e.g., Tactical French has over 50). We developed additional idioms to help us authors certain conversation patterns. For example, we use range-qualified transitions to implement a simple form of the "beat" concept as used in Façade (Mateas and Stern, 2005). This helped us more easily specify bits of dialog that can be used within certain phases of the story but do not directly change the story state beyond indirect effects such as increasing rapport. We are now designing a third generation of intent planning called LightPsych, which is an integrated framework where agents can operate at any of four levels – self (explicitly model beliefs, short- and long-term memory, etc.), culture (model and reason with politeness, and social norms), physical and social environment and dialog (understand dialog openings and closings, requests, turn-taking, negotiations, and the meaning of utterances). A key challenge for LightPsych is to make it easy to create and reuse agent-behavior specifications and handle the most common interactions between trainees and NPCs, but also give authors tools to add to and modify these specifications to handle the aspects that are unique to each scenario.

A critical issue is the representation of rich communicative acts. We started representing acts as strings that encoded some of the parameters needed by PsychSim agents. However, to accurately represent the richness of conversational speech, we need a richer representation for speech acts. We looked at frameworks such as FrameNet (Baker, C., Fillmore, C. and Lowe, 1998) before creating our own ontology of communicative acts based on the typology proposed by (Traum and Hinkelman, 1992) and extended with new acts (e.g., offering greetings, thanks, support etc.) based on the work by (Feng, Hovy, Kim, & Shaw, 2006). We also added representations of conversational context, such as the characteristics and roles of interlocutors, and the state of the world in which the conversation takes place. Our new act representation format extends the Functional Markup Language (FML) but extends it in a number of ways (Samtani, Valente and Johnson, 2008).

With respect to the generation of animated behavior, we have found that our initial hand-coded solution was optimally flexible but hampered our efforts to lower costs and speed development. Further, these scripting or motion capture approaches may work if the social context is fixed and known ahead of time, but break down if the social environment is dynamic. Therefore, we started working on automating the production of behavior animation from intent, and are exploring the use of representations such the Behavior Markup Language (BML).[6]

Authoring

Our authoring tools are designed to help authors create the rich content representations required by our AI-based system, and perform AI-based processing themselves. For example *Hilo*, our lesson authoring tool, supports rich utterance representations for speech and natural language processing. Utterances are modeled in different channels – the native orthography of the foreign language, an "ez-read" transliteration that is intended as a phonetic transcription in Roman characters, a phonetic transcription used by the speech recognizer, and a translation in English. Each utterance is also linked to the language model library so it can be centrally managed. Per-language tools are provided to support creation of some channels based on others. For example, a tool is provided for authoring French that proposes phonetic transcriptions for utterances written in standard French orthography.

Our *Tide* authoring tool provides several editing functions and analysis tools to specify interactive dialogs. Authors specify the utterances that may arise in the course of a conversation, their meaning in terms of acts, and the possible interactions between acts in the conversation. *Tide* supports several dialog models, from simple scripts to branched stories to agent-centric models. It provides graphical tools with a custom vocabulary of abstractions to make interaction authoring as easy as possible. It automatically generates program code from the specifications, which can be incorporated into the TLCTS clients. Finally, *Tide* provides a model checking tool for testing dialog modeling codes against the specification. This makes it possible for authors to produce a substantial portion of the dialog modeling work themselves.

Our *Hua* tool manages the language model, which contains the library of words and utterances that are taught in the course. It interoperates with *Hilo* and *Tide* as needed so that as authors edit content they can maintain a consistent language model throughout. This helps to eliminate errors, and reduces the role of natural language processing specialists in creating and maintaining linguistic resources.

These authoring tools have enabled us to increase the quality and quantity of authored content, and have made it possible to maintain multiple versions of content for different

---

[6] http://www.mindmakers.org/projects/BML.

classes of learners and different platforms. Table 1 shows some of productivity improvements that these tools are yielded. It compares *Tactical Iraqi* v3.1, developed in 2006, *Tactical Iraqi* v4.0, developed in 2007, and *Tactical French* v1.0 also developed in 2007. The authoring tools doubled the number of lesson pages, vocabulary words, dialogs, and Mission Game scenes in *Tactical Iraqi* v4.0 vis-à-vis v3.1. Development of *Tactical French* v1.0 did not start until late 2006, yet it has about the same amount of Skill Builder material as *Tactical Iraqi*.

Content Metric	TI 3.1	TI 4.0	TF 1.0
Lessons	35	52	40
Lesson pages	891	2027	1920
Words	1072	2214	1820
Example dialogs	42	85	67
Active dialogs	13	29	36
Scenes	8	18	9

Table 1: Content size of TLCTS courses.

Learner Modeling

As learners use TLCTS courses, it is important to track whether the learners are making progress toward learning objectives. Evidence for progress can come from multiple sources: the trainees' performance on games and quizzes, their performance in dialogs, and their performance generally in the TLCTS games. TLCTS records the learners' performance in quizzes and dialogs, but also attempts to estimate the learners' mastery of key skills, using a technique inspired by the model tracing technique of Corbett et al. (1995). Authors annotate exercises, quiz items, and dialog exchanges to indicate the skills that they employ. Skills are drawn from a taxonomy of Enabling Learning Objectives, encompassing linguistic skills, cultural skills, and task skills. Each correct or incorrect use of a given skill provides probabilistic evidence of mastery of that skill. This evidence is uncertain because learners may guess an answer, slip and make an unconscious mistake, or the speech recognizer may misinterpret the learner's response. However, after a relatively short amount of training time the learner model is usually able to correctly identify the individual skills that the trainee has mastered.

This learner model provides learners, trainers, and researchers with a rich view of trainee progress. We plan to use it as a basis for automated provision of training guidance, to advise learners about where they should focus their efforts, and when to adjust the amount of scaffolding to increase or decrease the level of challenge of the game. We also plan to use it as a starting point for estimating learner progress toward achieving longer term learning objectives such as general spoken language proficiency.

Application Use and Payoff

Tactical Iraqi was first deployed in June of 2005. Since then, several expanded and improved versions have been released, and additional courses have been developed. Courses are available in multiple versions for different user communities: for US Marines, US Army, non-US military forces, and civilian aid workers.

The courses are used by individuals training on their own, and as part of organized training programs. Anyone with a .mil email account can register and download copies either for their own use or for installation in computer labs. In January 2008, a typical month, there were 910 downloads of Tactical Iraqi, 115 downloads of Tactical Pashto, and 146 downloads of Tactical French.

Patterns of usage depend upon availability of training time and suitable computers. Many learners who download copies for their own use study them to completion; depending upon the course, this can require 100 or more hours of training. For military unit training, 20 to 40 hours of training are more the norm due to conflicting demands on training time. Some units combine TLCTS training with classroom training, whereas others rely exclusively on TLCTS for their language and culture training.

Several independent evaluations of Tactical Iraqi and Tactical Pashto have been performed by the US military, the Canadian Forces, and the Australian Defence Force. Results from several rigorous evaluations have been reported in (Surface & Dierdorff, 2007) and (Johnson & Wu 2008). Surface et al. studied several subject groups: 268 military advisors training at Ft. Riley, KS, 113 members of the 7th Marine Regiment, and 8 trainees at the Defense Language Institute Foreign Language Center (DLIFLC). All groups trained for a total of 40 hours, either with Tactical Iraqi alone or a mixture of Tactical Iraqi and classroom instruction. All showed significant gains in knowledge of Arabic language and culture, and greater self-confidence in communicating in Arabic. The greatest learning gains were achieved by the DLIFLC group, which trained exclusively with Tactical Iraqi and followed our recommended program of instruction. Six out of 8 participants achieved an ILR proficiency level of 0+ after 40 hours of training.

The marines in the 7th Marine Regiment are currently subjects of a Marine Corps Lessons Learned study. The 2nd Battalion, 7th Marines (2/7 Marines) and the 3rd Battalion, 7th Marines (3/7 Marines) returned from their most recent tour of duty in Iraq in December of 2007. Each battalion had assigned two members of each squad to undertake 40 hours of Iraqi Arabic language and culture training prior to deployment. The 2/7 Marines followed a blended training program, while the 3/7 Marines trained solely with Tactical Iraqi. Both battalions had successful tours of duty. The experience of the 3/7 Marines was particularly noteworthy because the battalion did not suffer a single combat casualty during its tour of duty. To understand the role of Tactical Iraqi in the 7th Marines' success, members of the two battalions were asked to complete questionnaires and the officers in charge of the 3/7 were interviewed. The ques-

tionnaires are still being tabulated, but transcripts of the officer interviews are available, and their comments are strikingly positive. The marines who trained with Tactical Iraqi were able to perform many communicative tasks on their own, without reliance on interpreters. This enhanced the battalion's operational capability, enabled the battalion to operate more efficiently, and resulted in better relations with the local people. This provides indirect evidence that Tactical Iraqi contributed to a Kirkpatrick level 4 training effect (i.e., impact on job performance) (Kirkpatrick, 1994). Follow-on assessments of the 3/7 Marines' language retention are planned for this year.

Future Work

We continue to improve TLCTS based on experience gained with earlier versions of TLCTS courses. Current work includes broadening the architecture to support reading and writing skills and deepening the curricula and platform to help learners attain higher levels of language proficiency. Language proficiency training is particularly challenging because it helps trainees get to the point where they can construct and understand new sentences in the target language, which complicates speech processing.

We continue to develop our Web-based *Wele* client as an option for learners with less powerful computers. The increased ease of installation and use will hopefully compensate for the reduced level of 3D rendering and interaction that will result from the constraints of web applications.

We are adapting TLCTS virtual-human technologies so they can be integrated into mission rehearsal training simulations. The training simulations will be populated with non-player characters that speak the target language. This poses new challenges for authoring since military trainers with no special technical training will create their own virtual worlds and populate them with non-player characters.

We continue to develop new courses, and develop pilot versions of future courses. A pilot Chinese course for college and high school Chinese programs is currently being developed in collaboration with Yale University Press and is scheduled to undergo classroom testing in the summer of 2008. A pilot Cherokee game, intended to help preserve Native American language and culture, is being developed in collaboration with Thornton Media, Inc.

Acknowledgments

References

Baker, C., Fillmore, C. and Lowe, J. 1998. The Berkeley FrameNet project. In Proceedings of the COLING-ACL.

Corbett, A., & Anderson, J. R. 1995. Knowledge tracing: Modeling the acquisition of procedural knowledge. User Modeling and User-Adapted Interaction, 4, 253-278.

Feng, D., Shaw, E., Kim, J., and Hovy, E. 2006. Learning to detect conversation focus of threaded discussions.

Johnson, W.L. & Wu, S. 2008. Assessing aptitude for language learning with a serious game for learning foreign language and culture. Proceedings of ITS 2008, in press.

Kirkpatrick, D.L. 1994. Evaluating Training Programs: The Four Levels. San Francisco, CA: Berrett-Koehler.

Mateas, M. and Stern, A. 2005. Structuring Content in the Façade Interactive Drama Architecture. AIIDE 2005.

Meron, J., Valente, A. and Johnson, W.L. 2007. Improving the Authoring of Foreign Language Interactive Lessons in the Tactical Language Training System. SLaTE 2007.

Mote, N., Johnson, W.L., Sethy, A., Silva, J., & Narayanan, S. 2004. Tactical Language detection and modeling of learner speech errors; The case of Arabic tactical language training for American English speakers. InSTIL / ICALL Symposium, Venice, Italy.

Samtani, P., Valente, A. and Johnson, W.L. 2008. Applying the SAIBA framework to the Tactical Language and Culture Training System. AAMAS 2008 FML workshop.

Si, M., Marsella, S.C., & Pynadath, D. 2005. THESPIAN: An Architecture for Interactive Pedagogical Drama. Proceedings of AIED 2005. Amsterdam: IOS Press.

Surface, E. & Dierdorff E. 2007. Special Operations Language Training Software Measurement of Effectiveness Study: Tactical Iraqi Study Final Report. Special Operations Forces Language Office, Tampa, FL.

Traum, D. and Hinkelman, E. 1992. Conversation acts in task-oriented spoken dialogue. Computational Intelligence, 8:575–599.

Vilhjalmsson, H. and Marsella, S., 2005. Social Performance Framework. AAAI Workshop on Modular Construction of Human-Like Intelligence.

Glossary of Hawaiian terms

Kona	big island	Honua	world
Hilo	a Polynesian navigator	Keaka	theater
Hua (hua'olelo)	word	Wele (puna welewele)	(spider) web
Waihona	library	Uku (ukulele)	flea
Lapu	ghost	Paheona	art
Huli	search	Hoahu	warehouse
Kahu	administrator		