Journal of Mobile Multimedia, Vol. 7, No.1 (2011) 030-050

PERVASIVE LANGUAGE LEARNING ON MODERN MOBILE DEVICES

BARTHOLOMÄUS WLOKA

University of Vienna, Vienna
bartholomaeus.wloka@univie.ac.at

WERNER WINIWARTER

University of Vienna, Vienna
werner.winiwarter@univie.ac.at

Received March 5, 2011
Revised March 18, 2011

In this article, we describe our experience with deploying our previous work on machine translation and language learning on mobile Internet platforms, i.e. smartphones. We present UTROLL – a Ubiquitous Translation and Language Learning Environment, implemented on the Nokia N900 with Maemo5 and KANTEAM – KAnji TEAcher Mobile, implemented on the Samsung Galaxy Tab with Google's operating system Android. In the process of implementation, we have analyzed both platforms as ubiquitous learning devices with special focus on sensor and hardware capabilities as well as usability. This work is combined with our previous efforts and creates a bridge between a server-based machine translation system and an everyday smartphone user. We present a detailed description of both applications, UTROLL and KANTEAM, while comparing their capabilities with respect to their hardware and operating system issues.

Key words: Mobile computing, computer assisted language learning, machine translation

1 Introduction

Technical advances of mobile computing technology enable us to carry out complex, computer-aided tasks virtually anywhere. *Computer-Assisted Language Learning* has been greatly influenced by this development, and is moving from a static desktop setting to a *pervasive* mobile scenario. Especially language learning endeavors require a tremendous amount of time and the learner is confined to his desk or computer at home or at university. In our research, we aim to create a *ubiquitous* application environment, which not only enables users to constantly repeat learned vocabulary, but keeps track of their progress and provides a customized learning experience based on the analysis of their location and environment. The level of embeddedness in computer-assisted learning can be depicted with the help of a biaxial graph figure 1. As we add embedded components to a desktop computer, shown in the lower left corner of the diagram, we move towards the upper left corner, where we approach a pervasive system. Such a learning environment is characterized by its situational awareness, while still

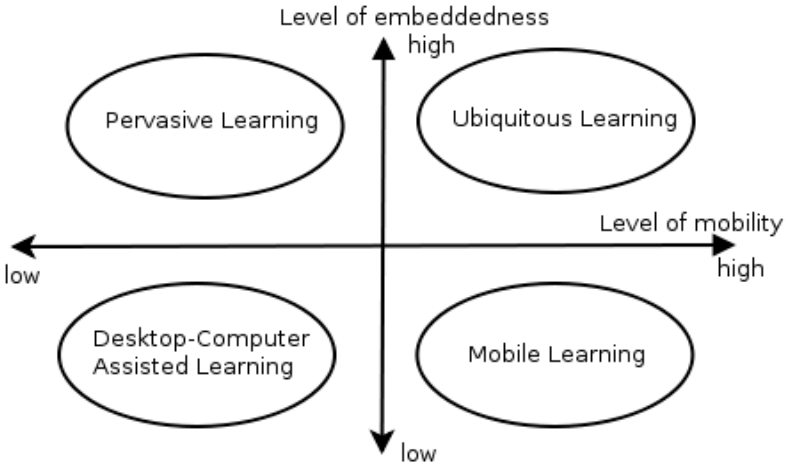

Figure 1 Types of learning environments [15]

being bound to a certain place. If we add a mobile component to the learning scenario, we move towards the right side of the diagram, where we have mobile learning in the lower and ubiquitous learning in the upper right corner. Our aim is to move as far as possible towards ubiquitous applications, where the users have access to learning materials anywhere and anytime, enabling them to benefit from a situationally aware system, which enhances the learning experience by adapting to the location, context, environmental conditions, and skill level of the user. This can be achieved by connecting a user-specific profile, following a learner-centered approach [13] with various sensor data obtained from the mobile device. This way, the learning application can interweave with almost every aspect of everyday life and accompany language students, hereby allowing for a constant progress and a language learning experience tailored to their needs.

Motivated by this, we have created a system architecture for a ubiquitous language learning and translation environment and, as a part of it, implemented prototypes of a mobile environment for the study of Japanese characters (*kanji).* Kanji are Japanese characters, adopted from China roughly 1,500 years ago. Due to their origin, these characters have at least a traditional Chinese and a Japanese pronunciation, and often several readings, depending on the context or the combination with other characters. There are several thousands of these characters in the Japanese language, from which roughly two thousand are commonly used. The different pronunciations, depending on the context, make them a key component in the complexity of learning the language. In addition to the characters, Japanese contains two syllabaries, *hiragana* and *katakana*, which can present the pronunciation of the kanji in a way similar to the Roman alphabet (*romaji*). Hiragana is mainly used for particles, suffixes, and inflections, katakana for transcribing foreign language words. This complexity of the learning material inspired us to begin our implementation work with KANTEAM and UTROLL implemented on the Nokia N900 and Samsung Galaxy Tab respectively.

Both programs offer a way to browse through pages of Japanese signs, similar to flashcards, but also the capability of reading sample sentences, which are contextually related to a particular word. A selection of these sentences can be stored in a personal database for later review. A pictographic example of how to draw the characters, which is vital for their correctness, is displayed through so

called *stroke order diagrams*. An appealing and intuitive design motivates the user to take advantage of the system and ease the burden of learning their way around a complex set of menus. Furthermore, the progress and the queries of the learner are sent to a knowledge base on the server and are stored for an analysis of the skill level. This functionality offers an opportunity for the instructor to manually review the strengths and weaknesses of the language student. Given a sufficient amount of students using the system, a collaborative scenario can be implemented by comparing learners' information on the server.

After a short discussion of related work in section 2, and our previous work on TREF (Translation Enhancement Framework) in section 3, we describe the architecture of both ubiquitous language learning and translation environments in section 4. Afterwards, we present showcases in section 5 guiding the reader through a series of screenshots, demonstrating the look and the capabilities of the applications. Following this, we compare the implementation on both systems in section 6 and present our findings as to how well these devices are suitable for the challenge. Finally in section 7, we conclude by summarizing our findings and giving an outlook on future work.

2 Related Work

One of the key components for our work are the language resources. We have chosen the *JMDICT* [3] and *KANJIDIC2* [4] resources published under the Creative Commons Attribution-ShareAlike Licence (V3.0). JMDICT consists of 137,000 Japanese head words with English glosses. KANJIDIC2 contains over 13,000 kanji entries with information about their readings, translations to English and other languages, their frequency-of-use ranking, and miscellaneous information about notation, encoding, indexing, etc. This information is structured in an XML file, which makes it ideal for our application. We have used both resources with slight alterations due to limited storage and computational capabilities on the mobile device.

The presentation of the learning material is equally important as the content itself [1], hence we have focused on a clean and efficient GUI design, considering the *cognitive load theory* [19]. We have followed the rule that a good balance between challenging the learners' focus and memory with content, and a sparse representation leads to a successful learning experience. Further, as confirmed by [7], contextual information, i.e. information inferred from the location and the learners' activities should be utilized to suggest meaningful learning content to the learner anywhere at any time [8].

[6] formulated a well-structured list of characteristics, which an application for ubiquitous learning should include in order to be successful.

- Accessibility: Learners have access to their documents, data, or videos from anywhere. That information is provided based on their requests. Therefore, the learning is self-directed.
- Immediacy: Wherever learners are, they can get any information immediately. Thus, learners can solve problems quickly. Otherwise, the learner can record the question and look for the answer later.
- Interactivity: Learners can interact with experts, teachers, or peers in the form of synchronous or asynchronous communication. Hence, the experts are more reachable and the knowledge becomes more available.

- Situation of instructional activities: The learning should be embedded in our daily life. The problems encountered as well as the knowledge required are all presented in their natural authentic forms. This helps learners to notice features of the problem situations that make particular actions relevant.

An example of the application of those guidelines can be found in [1]. The idea behind it is to give the students the chance to efficiently use their time and the ability to access class room information at will. In [10] a ubiquitous learning environment was developed by using IEEE 802.11 WLAN and Bluetooth for network communication. This showed that a learning experience, supported by a contextually matching surrounding, is more valuable in terms of understanding and memorizing since it is based on an inductive process. However, the limitations of the network technologies do not allow a deployment of that system in a large network structure.

As pointed out by [17], language learning is a life-long activity, and support by ubiquitous learning environments can accompany the learner at all stages. Language learning takes place virtually anywhere and is optimized if supplemented on demand. The need for immediate help makes translation resources on mobile devices very valuable, which is the reason why we have focused on combining those two, so that they complement each other in the best way possible. In the research area of intelligent tutoring [18], it was pointed out that the use of multiple representations delivers better results than, for example, repetition only. The use of pictures, as representations of objects, which were in the focus of the tutoring applications, proved to be very useful and enhanced the learning experience significantly. We have applied this idea in our work and present the learning contents graphically, as well as through several different representations. Furthermore, there is need for an efficient and clear user interface design [1]. This is a vital point, considering the vast amount of information available on a mobile device, combined with limited output capabilities, e.g. a small display. [5] has addressed this issue in the CAMELEON project, proposing a dynamic user interface. According to the cognitive load theory [19], the amount of information presented to the learner has a critical effect on the outcome of the learning process. Therefore, it is an important design consideration while building a learning environment. Only information which is stored by the short term memory can be processed and then stored in long term memory. If the student is faced with too much information at once, some of it might never be properly processed and is therefore lost. Too little information can lead to an “idle” state of the short term memory, which makes it less productive. Hence, it is vital to keep adequate levels of information presentation, depending on the students’ abilities as well as the situation. The abilities are defined by how advanced they are in the learning process, which can be monitored by short quizzes. The situation, in this context, is monitored by sensors, such as GPS and microphone, which measure how well the students can focus on their study at any given time. Research efforts by [7] show that using contextual information, i.e. information inferred from the location and the activities should be associated with the learning context to present a meaningful learning content.

In our previous work we have focused on knowledge-enhanced *statistical machine translation* which produced good results for the English-Japanese language pair. Though there are already many different approaches in machine translation [22], we have achieved an improvement with a hybrid approach, combining *relational sequence alignment* from bioinformatics [11] and statistical machine translation. We have found that our *TRanslation Enhancement Framework, TREF* [23], which uses templates from sequence-aligned and clustered sentence pairs, assumed to be similar, is a feasible way

of improving translation candidates from statistical machine translation systems. Since our method is both *example-based* and *corpus-based*, therefore containing a great amount of intermediate information from the translation process, it is well applicable in a language learning context. In the following section, we provide some background information on TREF, which is necessary for the understanding of the rest of the paper.

3 Translation Enhancement Framework

Machine translation between natural languages, which differ significantly in surface characteristics (this applies to most European-Asian language pairs), is a difficult task, and existing approaches do not produce satisfying results. With TREF we have succeeded to make a step forward in this area. Therefore, we applied these results in both our implementations on mobile devices. The underlying assumption of TREF is that there is a significant overlap between the structure of a sentence and its meaning. Based on this assumption TREF uses a sequence alignment algorithm by [11], taken from the field of bioinformatics, to make decisions about the restructuring of a translated sentence. As a translation basis we take the output of the statistical machine translation system Moses [9]. The overview of the dataflow in TREF is shown in figure 2.

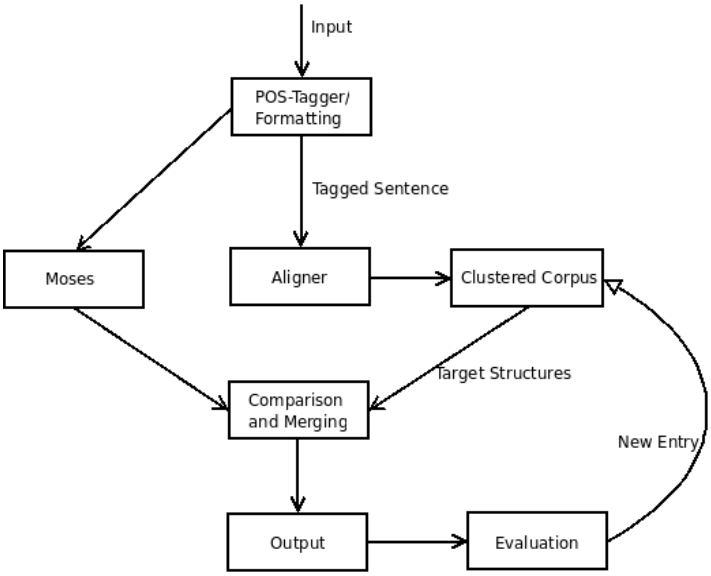

Figure 2 Translation enhancement framework dataflow

An input sentence is sent to the part-of-speech (PoS) tagger MontyLingua [14] for English, and ChaSen [16] for Japanese. After each sentence token is assigned a PoS-tag, the sentence and its tags are compared with sentences from a preformatted corpus. For this purpose, we have modified and enriched the Jenaad Corpus [20], a bilingual data collection consisting of 150,000 sentences, taken from news articles. We have removed as much noise as possible from the data, assigned PoS-tags to

each sentence token and stored the information in an SQL database. We have created different formats of the bilingual data, one with a complete set of PoS information and others with reduced and optimized tag sets to provide quick access and efficient processing. Additional representations and tag sets can be added easily to satisfy different needs in future work. We have applied relational sequence alignment [11] to obtain clusters of structurally similar sentences. The comparison of the query sentence with the clusters yields several similar structures. At the same time, the query sentence is processed with Moses to obtain a preliminary translation. This translation is then used to fill the template of the structures, which had been found to be similar in terms of PoS-tags. This way, a certain number of translation candidates are produced. The parameters of the similarity measure can be adjusted to fine-tune the result, depending on the text type and text domain. Allowing low threshold values for similarity, a higher number of candidates can be produced, whereas a higher threshold value reduces the number of candidates.

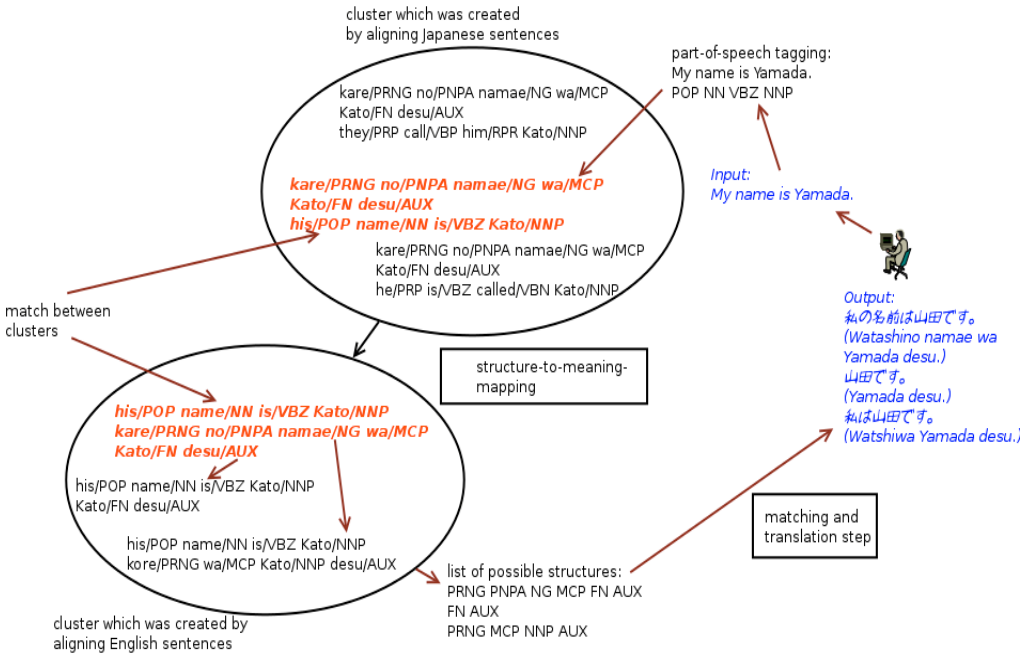

Figure 3 TREF processing example

The example in figure 3 illustrates how the sentence “My name is Yamada” is processed to present the user with the output, i.e. the translation candidates. The Japanese words in this schematic representation are written in Roman transcription. The acronyms represent the PoS-tags from MontyLingua and ChaSen, e.g. NNP (proper noun singular) or VBZ (verb, 3rd person singular present). First, the PoS-tagging of the input sentence is performed, in this case: My/POP (personal pronoun), name/NN (noun), is/VBZ (verb, 3rd person singular present), Yamada/NNP (proper noun). The alignment detects sentences in the database which are similar in terms of words and PoS-tags. In this step, different weights can be assigned to word or tag matches, so the output can vary. We still

experiment with different parameter settings on large input data, to improve the alignment results. All structures of those similar sentences are considered structure candidates for the final translation. The two groups of sentences in figure 3 represent the concept of structure-to-meaning-mapping as mentioned before. The candidates obtained from this mapping are sent to the matching and translation step to improve the translation result from Moses. Once the translation is finished, the target sentence is sent back and displayed on the client. As a useful feature in terms of language learning, the user can choose to display the PoS-tagged sentence from the translation process to obtain an insight about the linguistic characteristics. In the future we plan to integrate a search for similar sentences based on the aligning algorithm to find similar sentences for study suggestions. The interface of TREF for the communication with the client is the Django Web framework (www.djangoproject.com). In our previous work we have used Django to build a Web site interface. This site provides the access to the TREF module and includes the translation functionality, PoS-tag output, random sentence output from the Jenaad corpus as well as legends for the PoS-tags of MontyLingua and ChaSen. The translation of the original Japanese ChaSen tags into English was part of our previous work and is, to the best of our knowledge, the only English ChaSen PoS-tag legend available. The entire content is accessible through the Web interface at http: //wloka.dac.univie.ac.at.

4 System Architecture

4.1 UTROLL

The application is designed as a client-server setup. The client is the cell phone, in our case the Nokia N900. The LAMP (Linux-Apache-MySQL-PHP) server is situated at the University of Vienna. The operating system on the N900 is the open-source Maemo5, running on a Linux kernel, designed for smartphones and Internet tablets. This platform was developed in collaboration with the Linux kernel, Debian, and the GNOME project. The GUI and application framework is Hildon, which was originally developed by Nokia and recently became a part of GNOME. For development, we have used Scratchbox, a cross-compilation toolkit (www.scratchbox.org/), in combination with Python and GTK+, a cross-platform widget toolkit for creating graphical user interfaces. In contrast to distributions for desktop computers, Maemo5 has touch screen support, sliding keyboard support, an interface to the camera and to other sensors, while discarding some typical desktop distribution functionalities. Even though the hardware on the client side is quite powerful for a cell phone, it is not enough to perform all the calculations needed for our framework in a reasonable amount of time. Hence, we need to outsource as many calculations as possible to the server, where they can be processed much quicker.

Another reason is the fact that the main part of the translation module, TREF utilizes SWI-Prolog [21], and as of now, there exists no Prolog compiler for the Nokia N900 architecture. It is unlikely that efforts will be undertaken to offer such a compiler any time soon, since even though SWI-Prolog is very efficient, the computational overhead created by Prolog does not make it a good language choice for mobile devices in general. We have considered the fact that a cellular phone is not always guaranteed to have a decent network connection. Spots without a carrier signal, such as tunnels, elevators, etc. have to be taken into consideration. Therefore, the communication between the server and the client is done over asynchronous calls, to guarantee a service even in the case of an interruption of the network connection. Additionally, a database on the mobile device is kept to store

user input, such as vocabulary lists or program preferences, and enable system use while the network and/or the server is not available. Outdated entries from this database are purged periodically, due to limited storage capabilities on the cell phone. The overview of the server-client architecture is shown in figure 4.

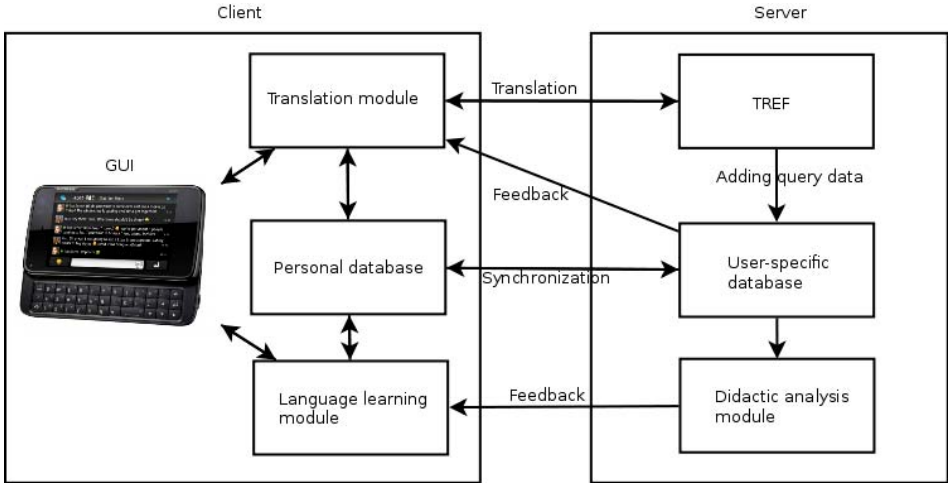

Figure 4 UTROLL – Server-client architecture

The translation module presents the contents graphically and provides input fields. The communication with the TREF module is initiated once a translation query is received. The language learning module offers a graphical learning program with several features. Data from the knowledge base on the server are used to construct a customized learning support, in terms of word/sentence suggestions and difficulty level. Both interfaces store essential data on the device. The language learning module accesses the personal database on the phone, which holds the necessary information to operate the framework without a network connection, though with limited capabilities. This personal database is synchronized with the user-specific database on the server, whenever possible. The user-specific database, as shown in figure 4, stores the user's history, such as previous query sentences. Words, compounds, and sentences stored in this database are assumed to be of interest to the student and are preferred in lecture suggestions. In future work, we plan to use ontologies in combination with the user's queries to find contextually fitting study suggestions. The didactic analysis module makes decisions about the difficulty and the representation of the learning data by analyzing the user data from the user-specific database. The user can also provide information about his environment, e.g. his degree of attention, similar to [1].

We also plan to automatically infer as much information as possible by monitoring sensor data, to perform situation analysis by using GPS and noise level data. In the future, we intend to integrate a voice recording module as well as a text capture module. The recording module will recognize words, spoken into the microphone of the cell phone, and translate and store them in the user's personal database. The text capture module will extract text from pictures, taken with the camera of the cell phone. The text in the picture will then be identified, translated, and stored in the personal database.

We have incorporated the server functionalities of our previous work, which are shown in figure 4. The client in this case is symbolized by the Nokia N900 device.

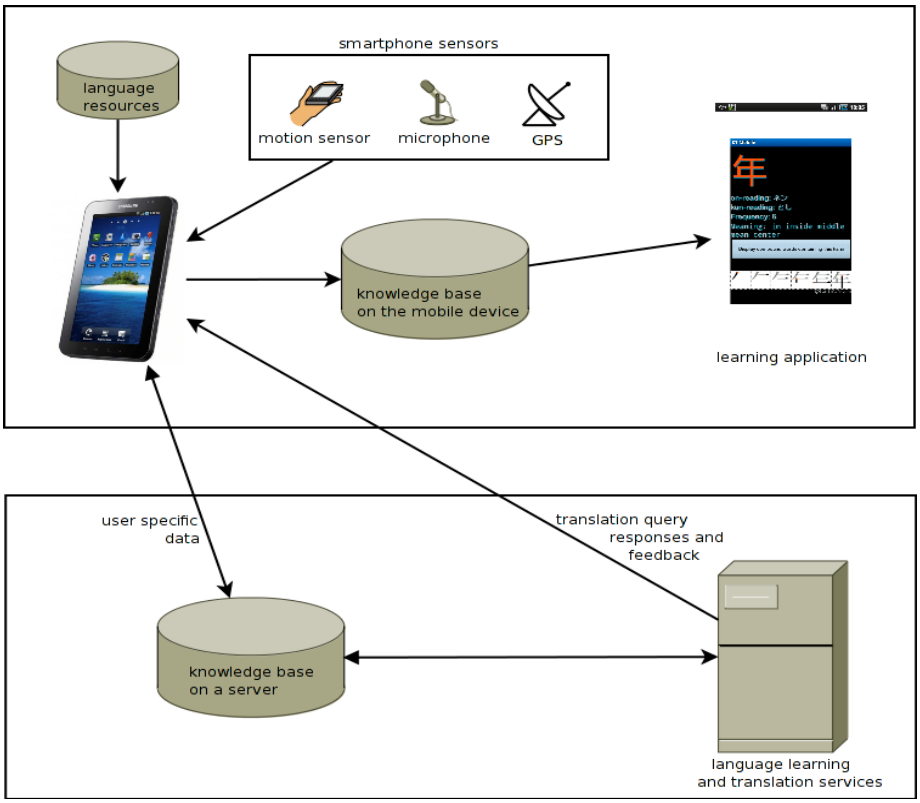

Figure 5 KANTEAM Server-Client Architecture

4.2 KANTEAM

The ubiquitous learning framework architecture is depicted in figure 5. The starting point is the Internet tablet or smartphone, symbolized here by a picture of the Samsung Galaxy Tab. The application on the phone is constantly fed with sensor data from the smartphone motion sensor, microphone, and the GPS. Upon startup of the *learning application*, this data, as well as user specific data from the server, if an Internet connection is available, is queried to present the learners with the best possible learning material for their particular needs. Whether the users choose to start a kanji lesson or type in a sentence to be translated, the action is forwarded to the server (or temporarily stored on the device in case the network is not available) and used to analyze the users' profile. On the server a *knowledge base* keeps track of the users' progress, location and environment data, which is used to infer the custom learning experience. The knowledge base *on the mobile device* serves a similar purpose, though not all data is located here due to obvious space and computational constraints. The *language learning and translation service* on the server offers translations and sentence analysis

queries through TREF. Further, we have incorporated *language resources* and a local *knowledge base* on the mobile device to guarantee basic functionalities in case the server is not available.

5 Showcase

5.1 UTROLL

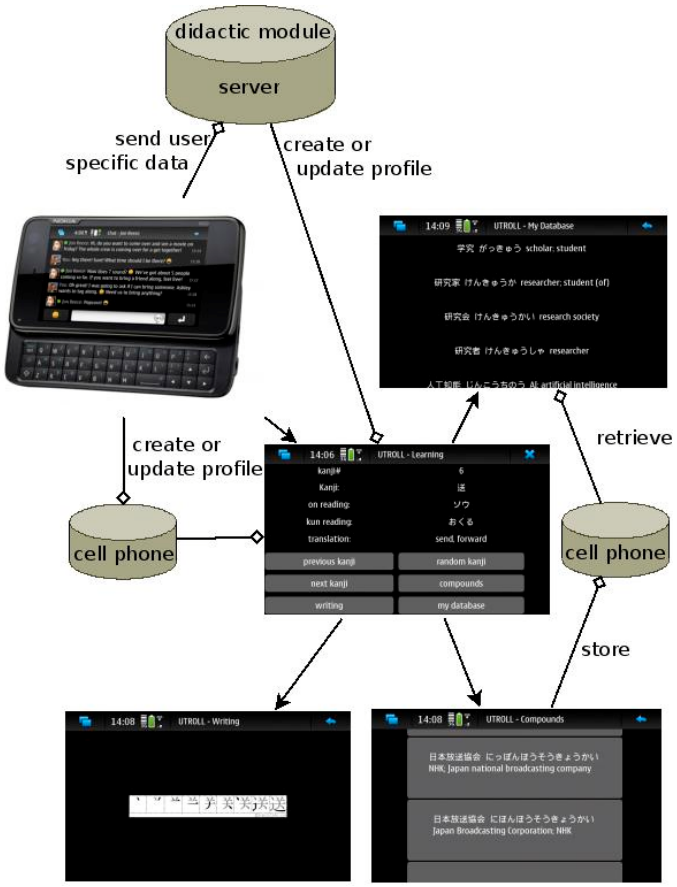

Figure 6 UTROLL – Language learning workflow

In this section we present the components of our Ubiquitous TRanslatiOn and Language Learning (UTROLL) environment. We have implemented a Japanese-English translation module, and a kanji learning module as part of the language learning task of UTROLL. The workflow of the kanji learning module and its GUI are demonstrated in figure 6. In the Starting View the user has the choice of starting the Translation Task or the Language Learning Task. Upon initiating the Language Learning Task, there is an attempt to connect to the server. If the connection is successful, the user-specific data is transmitted to the server, where the didactic profile is generated, according to the data in the user-specific database. The result, which is sent back to the phone, determines the difficulty level of the learning material. In order to maintain a challenging learning experience with growing skill, the

didactic analysis is performed periodically. In the kanji learning application, users can review flashcards with various information about a kanji, which they can choose to be displayed randomly or in a sequence suggested by the didactic profile. We call this the Flashcard View figure 7. From here, the student can switch to the Stroke Order Diagram View, as depicted in figure 8, where the exact way of drawing the kanji is presented with a sequence of images. We strictly follow the cognitive load theory mentioned before by separating the information into different views. The Compound View shown in figure 9 is a collection of compound terms, extracted from JMdict, which include the currently selected kanji. Users can store any of those compounds in a personal database. These stored compounds can later be reviewed in the Personal Database View (figure 10), which is accessible from the Flashcard View. The stored entries are considered in the process of creating or updating a didactic profile, in a way that the future suggestions will emphasize on similar terms or the domain of the terms.

Figure 7 UTROLL – Flashcard view

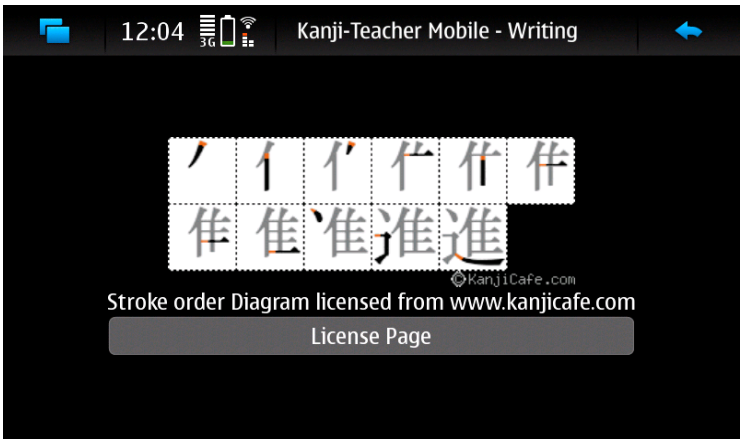

Figure 8 UTROLL – Stroke order diagram

Figure 9 UTROLL – Compound view

Figure 10 UTROLL – Personal database view

Selecting the Translation Task from the Starting View initiates a workflow as shown in figure 11. Upon initiating the task, a connection to the server is attempted. In contrast to the language learning task, a connection to the server is vital. In the first view (figure 12), the user has the choice of the input language. An input box (figure 13) takes the query sentence and sends it to the TREF module, which is located at the server. In the next view (figure 14), the translation result is displayed and the query sentence is stored on the server for later use by the didactic module. Additionally, the user has the choice of viewing further language details (figure 15). At the time of writing, our application displays various PoS-tag informations. In the future we plan to integrate a more detailed linguistic analysis, e.g. dependency trees produced by CaboCha [12]. Even though the current programming libraries available for the Maemo5 platform are fairly intuitive and easy to program, there is still a lack of certain design

capabilities, so that there has been a significant discrepancy between our initial design and the final realization. Unsightly gaps between the lines or between buttons are almost impossible to avoid using the current version of the SDK. The final implementation of the application on the Maemo platform is available for download at: http://maemo.org/ packages/view/ktmobile/.

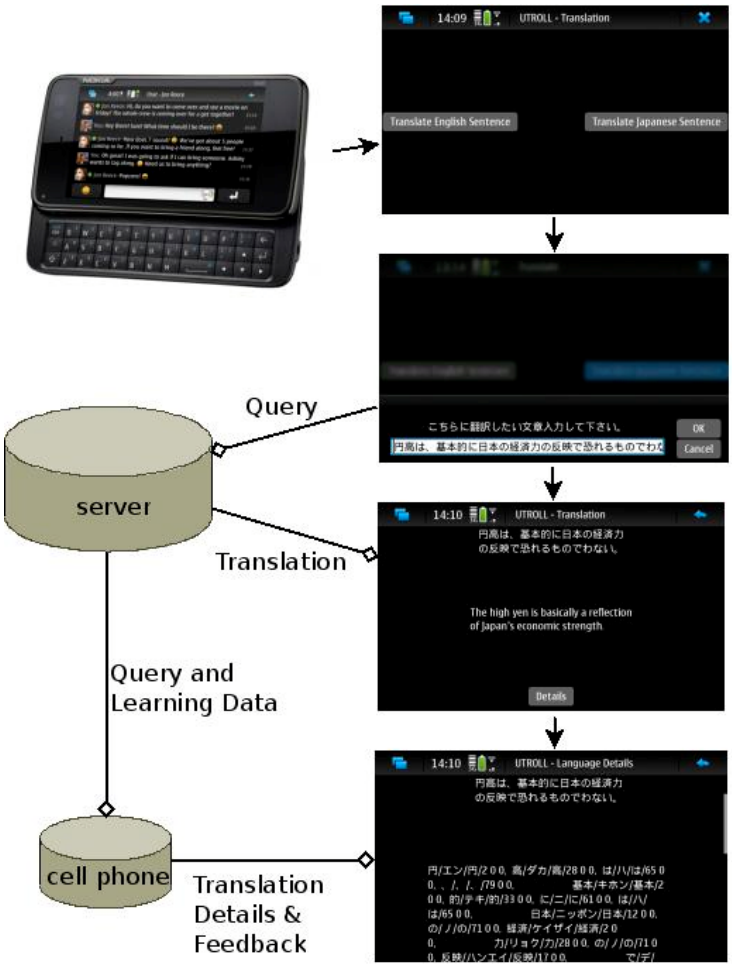

Figure 11 UTROLL – Translation task workflow

Figure 12 UTROLL – Language selection view

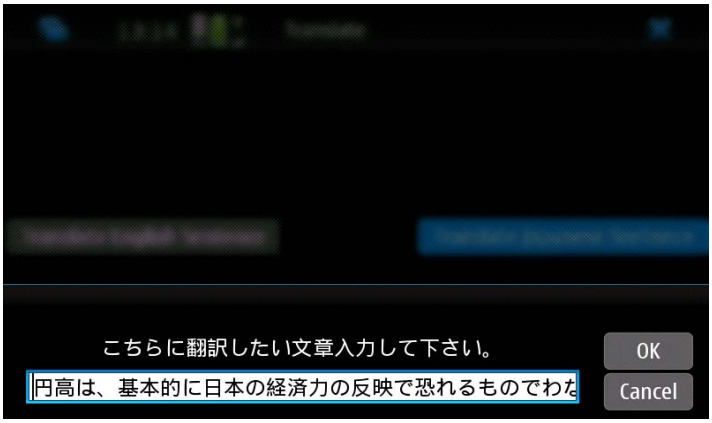

Figure 13 UTROLL – Sentence input view

Figure 14 UTROLL – Translation result view

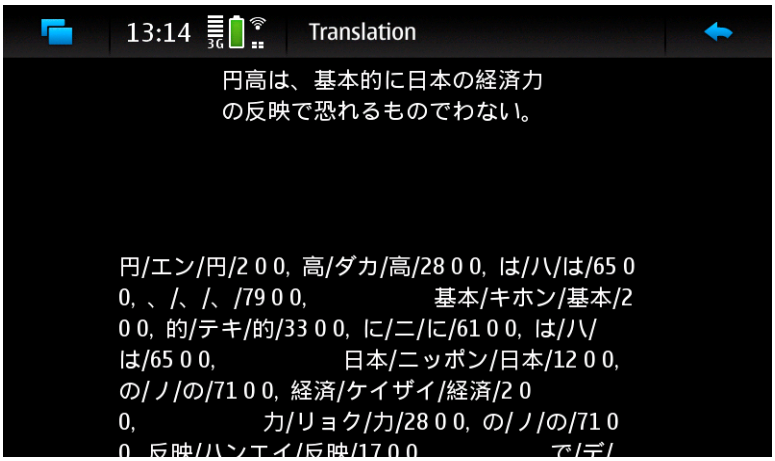

Figure 15 UTROLL – PoS-tag view

5.2 KANTEAM

In this section, we present a showcase of the Android 2.2 application KANTEAM. It is a flashcard-like application in which learners can choose to display kanji information based on different criteria, e.g. frequency-of-use of the character or a contextually inferred suggestion. The data can also be explored sequentially moving from screen to screen by finger-swipe or selecting a random entry by shaking the device. An example screenshot of a representation of the kanji data is shown in figure 16. Below the title of the application, the kanji is shown in red color. Further, the different readings of the kanji are denoted in katakana and hiragana script, followed by the frequency ranking and the English meaning. The button just below the meanings takes the learner to a dynamically created list of compound terms (combinations of two to four kanji and their meanings). At the bottom, the stroke order diagram describes the correct sequence of drawing this particular character. From the main menu, which is accessible through a button on the Android device, the user can choose to input a sentence, Japanese to English or vice versa, to translate via the server-based part of the system as seen in figure 17. Upon entering the query, the result is displayed as in figure 18 and the learner has the opportunity to take a detailed look at the *part-of-speech tagging* data of the translation process to understand the grammatical properties of both the input as well as the translated sentence (figure 19).

Figure 16 KANTEAM screenshot

Figure 17 Translation – sentence input

Figure 18 Translation – result

Figure 19 Translation – Part-of-speech tags

6 Discussion

After implementing our application on both systems, we have found that both offer a good platform for a server-client language learning application. The API and the interface to the hardware on both smartphones are well implemented, though the Android system seems to have a slight advantage due to its wider popularity. The GUI design was more intuitive and flexible on the Android system. However, text manipulation is easier and more efficient on the Maemo5 platform.

The implementation of the Android system was done in the Eclipse-3.5 software development framework, using the Android Developer plug-in provided by Google. The Android programming language, which is Java-based and heavily relying on XML-defined structures, proved to be an excellent choice for our purposes. These structures allow for quick and intuitive GUI design. This model is inspired by the Web development model where the presentation of the GUI and the application logic is separated. The communication between those two components is executed behind the scenes by the Android operating system. The Android system seems to have an advantage when it comes to efficiency while dealing with large amounts of data. It comes with a formidable library and interface to XML files, while this can be somehow cumbersome on the Maemo5 platform. The XML handling capabilities on the Android system enabled us to import the language resources very quickly and iterate over a large amount of data without any noticeable performance issues. Compared with the Maemo5 framework, the implementation was more straightforward, even though the support for the Python language on the Maemo5 device offered more flexibility when working with string manipulation.

As can be seen in the side-by-side comparison of software features in figure 20, the base of both systems is a Linux Kernel and both accept C/C++ as a language for developing applications. However, it is much easier to use Java on Android and Python on Maemo5, due to better library support and easier, more abstracted, source code. When it comes to the development environment, Maemo5 does not offer a comprehensive solution, however, it is possible to develop on the device itself if needed and when using Python as the language of choice, no compilation is necessary and the application can be tested instantaneously. Android on the other hand offers the Eclipse plug-in which makes porting the application to the device very easy.

	Android 2.2	Maemo5
Base	Linux 2.6	Nokia N900-50-1 Linux 2.6.28 - omap1 on armv71
Prog. Language Choices	Java, C/C++	C/C++, Python
Development Environments	Arbitrary, development can be done directly on device	Eclipse with plug-ins or any other (more complicated deployment using other)

Figure 20 Software comparison

A hardware comparison of the devices we have used is shown in figure 21. The Galaxy Tab has a small advantage over the slightly older N900. The presentation of apps profits greatly from the larger

screen on the Galaxy Tab. Due to its performance advantage and the more efficient data management of Android our application runs considerably faster on this device.

	Nokia N900	Samsung Galaxy Tab
CPU	ARM Cortex-A8 600MHz	ARM Cortex-A8 1.0GHz
Memory	256MB	512MB
Operating System	Maemo5	Android 2.2
HD	32GB	32MB
Screen	8.9cm diagonal, 800 x 480px	18cm diagonal, 1024 x 600px

Figure 21 Hardware comparison

Overall, the Android platform offers a visually more attractive solution. Since there are far more Android users and programmers than for the Maemo5 system, there is obviously more support for future work on Android. Other disadvantages of the Maemo5 platform are that, at the time of writing, it already seized its development to merge into the *MeeGo* project and the new partnership of Nokia with Microsoft has rendered the future of the open source system on Nokia devices very questionable.

7 Conclusion

In this article, we have presented the progress of our work on a ubiquitous language learning environment and an implementation of the client-side application on the Nokia N900 with Maemo5 and the Samsung Galaxy Tab with Android 2.2. We have described the system regarding ubiquitous language learning and its capability towards new research trends such as collaborative learning and utilization of smartphone sensor capabilities.

We are ready to deploy our prototypes to a classroom scenario at the language department of our university to obtain valuable feedback and evaluate the system regarding scalability as well as usability. We expect a wide community of users, especially for KANTEAM due to the popularity of the Android platform and its propagation to modern smartphone devices.

In the future, we plan to widen the ubiquitous component even more, by adding instructional scenarios based on the learners' location, in combination with Google's map and location services and add new and/or additional language resources, which are currently being developed, e.g. the Japanese *WordNet* [2], offering graphical illustrations of concepts for an even more valuable and intuitive learning experience.

Further, we would like to integrate the GPS capabilities of the mobile devices to offer a selection of courses, translations and language related suggestions to the user based on their current location and combine the current application with a graphical instructional adventure game, moving away from text-based lessons to visually enhanced presented learning material.

References:

1. Bomsdorf. B. (2005). Adaptation of learning spaces: Supporting ubiquitous learning in higher distance education. Mobile Computing and Ambient Intelligence: The Challenge of Multimedia, Dagstuhl Seminar, Dagstuhl, Germany, 2005.
2. Bond, F. et al. (2009). Enhancing the Japanese WordNet. 7th Workshop on Asian Language Resources, in conjunction with ACL-IJCNLP, 2009.
3. Breen, J. (2004). JMdict: A Japanese-Multilingual dictionary. COLING Multilingual Resources Workshop, 2004, ACL, Stroudsburg, PA. 65-72.
4. Breen, J.W. (2004). The KANJIDIC2 Project, http://www.csse.monash.edu.au/~jwb/ kanjidic2/
5. Calvary C. et al. (2002). Plasticity of user interfaces: A revised reference framework. In Proceedings of TAMODIA, 2002.
6. Chen, Y. et al. (2002). A mobile scaffolding-aid-base bird-watching learning system. IEEE International Workshop on Wireless and Mobile Technologies in Eduction, IEEE Computer Society Press, 2002. 15-22
7. Cui, Y. & Bull, S. (2005). Context and learner modelling for the mobile foreign language learner. System, 33(2), 353–367, 2005.
8. Gan, L.H. et al. (2007). Language learning outside the classroom using handhelds with knowledge management. Conference on Supporting Learning Flow through Integrative Technologies, Amsterdam, The Netherlands, 2007. IOS Press. 361–368
9. Hoang H. et al. (2007). Moses: Open source toolkit for statistical machine translation. pages 177–180, 2007.
10. Jones V. and Jo J. H. (2004) Ubiquitous learning environment: An adaptive teaching system using ubiquitous technology. In Proceedings of the 21st ASCILITE Conference, 2004.
11. Karwath A. and Kersting K. (2007). Relational sequence alignments and logos. pages 290–304, 2007.
12. Kudo T. and Matsumoto Y. (2002). Japanese dependency analysis using cascaded chunking. In CoNLL 2002: Proceedings of the 6th Conference on Natural Language Learning 2002 (COLING 2002 Post-Conference Workshops), pages 63–69, 2002.
13. Levy, M. (2009). Technologies in use for second language learning. The Modern Language Journal, 93, 769-782.
14. Liu H. and Singh P. (2004). Conceptnet — a practical commonsense reasoning tool-kit. BT Technology Journal, 22(4):211–226, 2004.
15. Lyytinen, K. and Yoo, Y. (2002). Issues and challenges in ubiquitous computing. Communications of the ACM, 45 (12), 62-65
16. Matsumoto Y., Kitauchi A., Hirano T.Y., Matsuda H., Takaoka K., Asahara M. (2000). Japanese Morphological Analysis System ChaSen version 2.2.1, 2000.
17. Ogata H. (2008). Computer supported ubiquitous learning: Augmenting learning experiences in the real world. In IEEE International Conference on Wireless, Mobile, and Ubiquitous Technology in Education, pages 3–10, Los Alamitos, CA, USA, 2008. IEEE Computer Society.
18. Rau M.A., Aleven V., Rummel N. (2009). Intelligent tutoring systems with multiple representations and self-explanation prompts support learning of fractions. In Proceedings of the 2009 Conference on Artificial Intelligence in Education, pages 441–448, Amsterdam, The Netherlands, 2009. IOS Press.
19. Sweller, J., & van Merrienboer J., & Paas, F. (1998). Cognitive architecture and instructional design. Educational Psychology Review. 10, 251–296, 1998.
20. Utiyama M. and Isahara H. (2003). Reliable measures for aligning Japanese-English news articles and sentences. In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics, pages 72–79, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

21. Wielemaker J. (2003). An overview of the SWI-Prolog programming environment. In F. Mesnard and A. Serebenik, editors, Proceedings of the 13th International Workshop on Logic Programming Environments, pages 1–16, Heverlee, Belgium, 2003.Katholieke Universiteit Leuven.
22. Wilks, Y. (2008). Machine Translation: Its Scope and Limits. Springer-Verlag, 2008.
23. Wloka B. and Winiwarter W. (2010). TREF – TRanslation Enhancement Framework for Japanese-English. International Multiconference on Computer Science and Information Technology, IEEE, 2010. 541-546