Personalized Reading Support for Second-Language Web Documents

YO EHARA, Graduate School of Information Science and Technology, The University of Tokyo
NOBUYUKI SHIMIZU, Information Technology Center, The University of Tokyo
TAKASHI NINOMIYA, Graduate School of Science and Engineering, Ehime University
HIROSHI NAKAGAWA, Information Technology Center, The University of Tokyo

A novel intelligent interface eases the browsing of Web documents written in the second languages of users. It automatically predicts words unfamiliar to the user by a collective intelligence method and glosses them with their meaning in advance. If the prediction succeeds, the user does not need to consult a dictionary; even if it fails, the user can correct the prediction. The correction data are collected and used to improve the accuracy of further predictions. The prediction is personalized in that every user's language ability is estimated by a state-of-the-art language testing model, which is trained in a practical response time with only a small sacrifice of prediction accuracy. The system was evaluated in terms of prediction accuracy and reading simulation. The reading simulation results show that this system can reduce the number of clicks for most readers with insufficient vocabulary to read documents and can significantly reduce the remaining number of unfamiliar words after the prediction and glossing for all users.

Categories and Subject Descriptors: H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Natural Language*; H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia—*User issues*; I.5.4 [**Pattern Recognition**]: Applications—*Text processing*

General Terms: Design, Experimentation, Human Factors, Languages

Additional Key Words and Phrases: Reading support, Web pages, item response theory, glossing systems, logistic regression

ACM Reference Format:
Ehara, Y., Shimizu, N., Ninomiya, T., and Nakagawa, H. 2011. Personalized Reading Support for Second-Language Web Documents. ACM Trans. Intell. Syst. Technol. 2, 3, Article A (September 11), 18 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

English is now so widespread that, in non-English speaking countries, office workers who are not specializing in English need to read English documents during their office work. Furthermore, more and more users are now browsing Web pages, using English word glossing systems, which eliminate the need to consult a dictionary every time an unknown word is encountered. A glossing system presents the user with the meaning of an unknown word in a pop-up window when the user clicks on or mouses over it. An example is "pop jisyo", shown in Figure 1. The background sentences that begin with "A group of Han Chinese" are sentences on a Web page that we assume the user is reading. Suppose that the user encounters the word "paramilitary" and does not know its meaning. If the user mouses over the word, a pop-up window opens and the meaning of the word "paramilitary" is displayed in it.

While useful, existing English glossing systems do not utilize the user's vocabulary. They waste valuable data about the user's vocabulary, which could be accumulated. Instead, we may harness collective intelligence by utilizing the accumulated word click logs from many users. We are devel-

Author's addresses: Y. Ehara, N. Shimizu and H. Nakagawa, The University of Tokyo, General Library 4th floor 7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-0033, Japan; e-mails: ehara@r.dl.itc.u-tokyo.ac.jp, shimizu@r.dl.itc.u-tokyo.ac.jp, and n3@dl.itc.u-tokyo.ac.jp; T. Ninomiya, 3 Bunkyo-cho, Matsuyama, Ehime 790-8577, Japan; e-mail: ninomiya@cs.ehime-u.ac.jp

oping a system that can accumulate this kind of knowledge and make full use of it to help users to read English Web pages by predicting the words that the user does not know and displaying their meanings [1]. Our aim is to combine an English word glossing system with word difficulty prediction in an integrated and intelligent user interface. Existing research on word difficulty relies heavily on corpus frequency. However, corpus frequency may not reflect the difficulty that users actually feel: some words occur frequently in corpora, but actually may not be known to the users, while some words are rarely seen in corpora, but actually may be known to the users. By accumulating the word click logs from many users and uncovering the "collective intelligence" beneath the logs, we can get the collective intelligence to adjust the weight for each word to reflect the difficulty that users actually feel; even if some words occur frequently in corpora, those words are regarded as difficult if many users frequently click them to see their glosses. Even if some words are rarely seen in corpora, those words are regarded as easy if many users click those words to hide their glosses. Thus, we can regard our system as using the collective intelligence embedded in the click log to correct the corpus-frequency-based word difficulty.

The main contribution of this study is a mechanism for predicting words unknown to a user. We use logistic regression to estimate which words on the Web page are unknown to the user because it is widely used to test language ability in "item response theory" (IRT). This theory lets one estimate both a user's language ability θ and a word's difficulty d at the same time. To train the logistic regression faster, we chose stochastic gradient descent (SGD) from among many parameter optimization methods because it is an online algorithm. An online algorithm can perform training faster than a batch method because it accesses a datum in the data set only once whereas a batch method accesses the data many times.

The example in Figure 2 shows how the system works when a reader, being an English learner as well, tries to read a Web page. The yellow words are predicted to be known to this reader and the red word, namely "lorries" in this example, is the word predicted to be unknown to the user. Thus, this red word is "glossed", that is, a gloss is attached to it. The system makes this prediction when the reader loads the Web page so that, from the reader's viewpoint, the Web page seems to come with the gloss in the first place. The reader can also mispredictions by clicking words: when the reader clicks a yellow (i.e., predicted to be known) word, the system is informed that the reader actually does not know the word. When the reader clicks a red (i.e., predicted to be unknown), word, the system is informed that the reader actually "knows" the word. These corrections are instantly dispatched to the system from the reader's Web browser by an Ajax script, so that the system can learn and reflect the correction in its prediction when the reader goes to another Web page.

Another contribution of this study is that we evaluated our system to see whether it can reduce the number of clicks and the number of unreadable documents. The t-test was performed for both evaluations and the results show that the system worked significantly well for these evaluations.

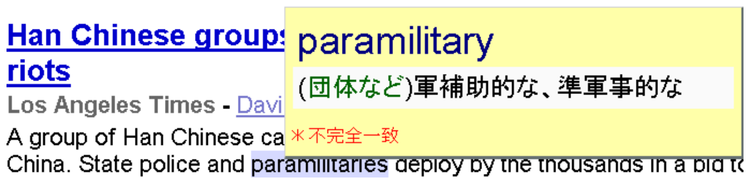
In summary, our system is characterized as follows.

(1) Accumulates knowledge about the vocabulary of each user and fully utilizes it
(2) Has a machine-learning-based mechanism for predicting words unknown to a user

This paper is organized as follows. First, we introduce related work. Second, we introduce the design and implementation issues of our system. We then explain the relationships among IRT, logistic regression, and the Rasch model, which is a special case of IRT and also a special case of logistic regression, which our system uses. We then explain the methods for estimating the parameters of logistic regression and the derivation of SGD. We then describe an experimental evaluation of our system. Finally, we conclude this paper by briefly summarizing the main points and mentioning future work.

Note that throughout this paper we use the term "unfamiliar words" to denote words unknown to a user. Although we know that word familiarity is, strictly speaking, a different concept from simple

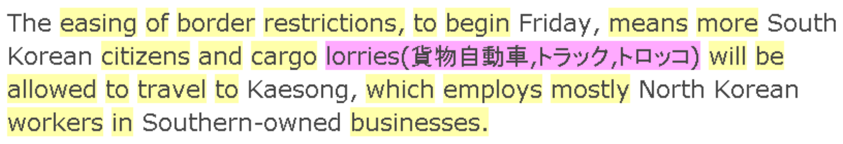
[1] `http://www.socialdict.com/`



Han Chinese groups
riots
Los Angeles Times - Davi
A group of Han Chinese ca
China. State police and paramilitaries deploy by the thousands in a bid t

paramilitary
(団体など)軍補助的な、準軍事的な
＊不完全一致

Fig. 1. Example of word glossing in pop-jisyo.



The easing of border restrictions, to begin Friday, means more South Korean citizens and cargo lorries(貨物自動車,トラック,トロッコ) will be allowed to travel to Kaesong, which employs mostly North Korean workers in Southern-owned businesses.

Fig. 2. Example of word glossing in our system.

word knowledge, since word familiarity is not the focus of this paper and the term “unknown words” has another meaning, we use “unfamiliar words” for simplicity.

2. RELATED WORK

This section describes previous work related to this paper.

Glossing systems for Web pages have been implemented in two forms: as desktop applications that retrieve Web pages and perform the glossing on the user’s local machine and as Web applications in which Web page retrieval and glossing is performed on a Web proxy server instead of on the user’s local machine. While desktop glossing systems are fast because they do not require communication between the Web proxy server and the user’s local machine, they are not suitable for accumulating the user click logs because the logs are distributed across many local machines in desktop applications. Since we accumulate and utilize the click logs, we implemented our system as a Web application.

Desktop glossing systems are usually implemented as add-ons for Web browsers, such as: **Fire-Dictionary** [Nori 2005], and **popIn** [popIn Inc. 2008]. They work as add-ons for the **Firefox** Web browser.

Notable Web-based glossing systems include **popjisyo** [Coolest.com Inc. 2002], which we used as an example, **rikai.com** [T. D. Rudick 2001]. In particular, **popjisyo** supports glossing between multiple languages [2].

However, as far as we know, no application estimates the users’ language ability or predicts the words unknown to the users from the accumulated click logs, though some applications may simply accumulate them for quick access to the words previously clicked. Our application is novel in that it assesses the user’s language ability and predicts words unknown.

Our approach also differs from machine translation in that our system glosses only those words predicted to be unknown to the user whereas machine translation does not utilize the user’s language ability and translates the entire text including words known to the user.

3. SYSTEM OVERVIEW

This section explains the operation of our system and implementation issues.

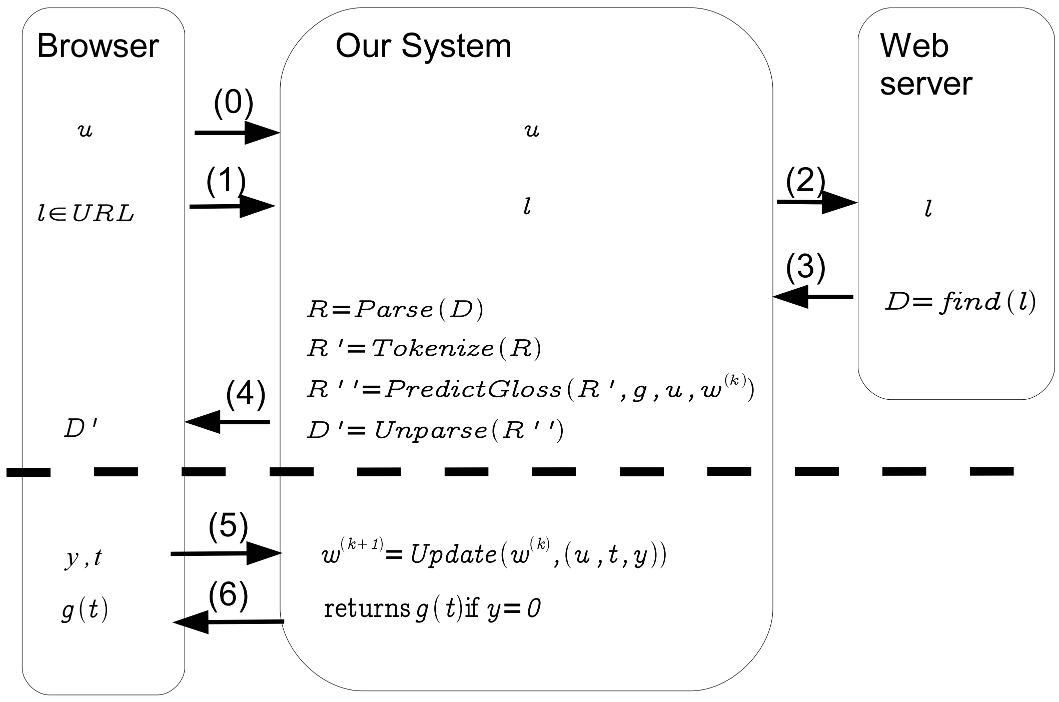


Fig. 3. System operation.

3.1. System Architecture

The way our system operates is schematically illustrated in Figure 3. It is a glossing system that uses “CGI-proxy” and predicts words unknown to the user by machine learning. In use, it operates as follows.

(0). The user passes user identifier u to the system. Note that a user identifier is not necessary in previous systems because they do not perform user adaptation.
(1). The user passes document identifier $l \in URL$ to the system.
(2). The system finds the Web server hosting the target document by following l.
(3). The system tries to retrieve document D specified by l from the Web server. If the system fails to retrieve the document, it simply returns an error message to the browser in step (4).
(4). The system extracts words from document D and returns the document with the words predicted to be unfamiliar to user u glossed with their meanings.

Web documents are usually have tags like “<html>” and “<body>”. Thus, they can be represented by a tree, as shown in Figure 4 (a). Let Dom_D be the set of all Web pages and Dom_T be the set of trees. We define $Parse(D)$ as a function that takes a Web page $D \in Dom_D$ and returns tree $R' \in Dom_T$, that corresponds to D. Conversely, we define $Unparse(R'')$ as a function that takes tree $R'' \in Dom_T$ and returns the Web page that corresponds to R''.

Step (4) in Figure 3 involves a tokenization function “Tokenize” and a prediction and glossing function “PredictGloss”. “Tokenize” takes tree $R \in Dom_T$ and returns a tree whose sentences are tokenized into words. An example is shown in Figure 4: “Tokenize” takes Figure 4(a) and returns

[2]Japanese to English, Japanese to German, English to Japanese, English to Korean, English to Spanish, Chinese to Korean, Chinese to English, and Korean to English.

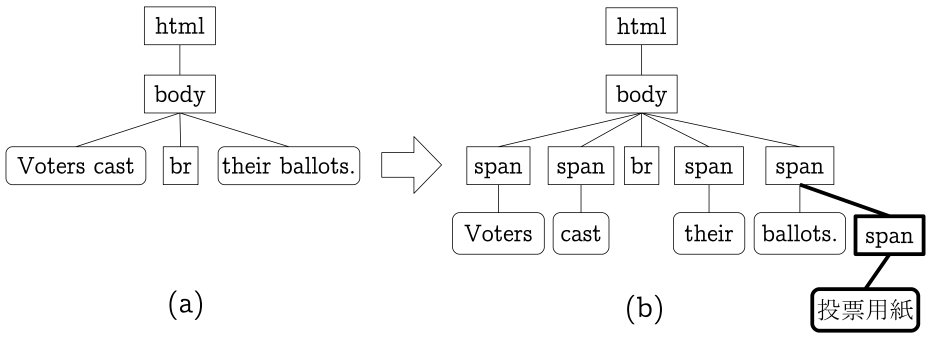


Fig. 4. (a) Example of a tree, (b) Tree passed back to the browser. The bold part is an example of a gloss, showing the meaning of ballots in Japanese.

the tree in Figure 4(b) except for the bold parts. "Tokenize" also tags every token by "<span>" and embeds a JavaScript script that enables the user to consult a dictionary simply by clicking the word.

"PredictGloss" takes tokenized tree $R' \in Dom_T$, glosser function g, user identifier u, and the weight of classifier $w^{(k)}$ and g takes token t and returns its gloss (meaning) $g(t)$. For all the leaves, i.e., tokens, in tree R', "PredictGloss" first predicts the words unfamiliar to user u and then glosses only unfamiliar ones with g. The prediction is determined by the sign of the function $h(u, t, w^{(k)})$.

While previous systems do not require any communication with the browser after (4), our system communicates with the browser in (5) and (6) to collect the "click log" (y, t) (defined below) to train $w^{(k)}$. This communication is made possible by the use of "AJAX", asynchronous JavaScript, and XML (extensible markup language). We used "jQuery"[3] for the library for AJAX.

(5). When token t in a different Web document D' is clicked, the embedded JavaScript script in D' sends the pair of y and t to the system.
(6). If $y = 0$, i.e., the user does not know word t, so its gloss is sent back to user u.

Here, y codes whether or not user u knows word t. It is defined as follows: $y = 0$ is sent to the system if word t was predicted to be unfamiliar to user u prior to the click and is clicked for the first time after this prediction. $y = 0$ means that the system regards user u as not knowing word t because user u is now requesting the gloss for word t even though the system first regarded user u as knowing word t in (4). If word t is predicted to be known to user u and is clicked for the first time, the exact opposite happens: $y = 1$ is sent to the system, which means that the system regards user u as knowing word t because user u is now hiding word t even though the system first regarded user u as not knowing word t in (4). The data (u, t, y) is used to update the value of $\boldsymbol{w}^{(k)}$ for training, as shown in Figure 3. The function $Update(\boldsymbol{w}^{(k)}, u, t, y)$ is explained later.

3.2. Implementation Details

First, note that the system should be multi-threaded and every thread processes the requests from one of the users. Simply speaking, our system involves two kinds of data: the weight vector $\boldsymbol{w}^{(k)}$ for classification, and the click logs accumulated so far to train the weight vector. These data have very different natures: the weight vector should almost always be stored in memory and accessible from every thread because it is used every classification. Moreover, it is frequently updated by the users' usage. By contrast, the click logs are, strictly speaking, not updated but rather added. Simply recording new click data suffices. Moreover, they do not need to be always loaded in memory as they are used only when the system retrains the weight vector, say during the night, for example.

[3]`http://jquery.com/`

Exploiting this different nature of the data, we store the weight vector in "memcached" and the clicks logs in a relational database management system (RDBMS). When the user clicks a word, the thread processing that user dispatches two requests: one to update the weight vector stored in "memcached" and one to add a new data record to the RDBMS. This "memcached" is the usual solution for this kind of environment; it is, simply speaking, a hash table shared by all the threads processing a user's request. Being a hash table—one update consists of the pair of a key (the id of a feature) and a value (the value of the feature)—means that "memcached" involves only that key and does not affect any other keys. This means that, for example, the English ability features of two different users can be simultaneously updated because they have different feature ids.

4. MODEL

Logistic regression is used to model our system because it includes the Rasch model, which is a basic IRT. IRT is widely used to test language ability. To predict the words unknown to a user, it is necessary to model the user's language ability. Thus, logistic regression is suitable for use in this application.

4.1. Item Response Theory

This section explains IRT and how it is involved in our system. IRT is a statistical method for analyzing the results of tests of human abilities including language tests. Indeed, IRT is thought to be used in TOEFL (Test of English as a Foreign Language) and TOEIC (Test of English for International Communication). Details of the mathematical aspects of IRT are given in [Baker and Kim 2004].

IRT has been used to generate fill-in-the-blank questions automatically from text retrieved from the Web [Sumita et al. 2005].

First, IRT calculates "parameters" from human ability test results. Once the parameters have been calculated, IRT can then estimate the following probabilities.

— probability that a user will answer a "different" question correctly
— probability that a "different" user will answer the original question correctly

Here, "different" means that no record with the same pair of user and question is found in the previous test results. This covers the user or question being a new one and the user or question existing but the user not answering that question in the previous test.

4.1.1. Definition of IRT. IRT involves three concepts: "users", "items", and "responses". Users are simply the set of users. Items correspond to the questions in a test. The response indicates how correctly the user responded to an item.

Given user $u \in U$ and item $t \in T$, Eq. 1 models user u's response y to item t, where U and T are finite sets of users and items, respectively, and y is the response given u and t.

Eq. 1 codes y as a binary random variable (though in some variations of IRT, y can take more than two states); that is, $y = 1$ when user u answers item t correctly and $y = 0$ otherwise. Eq. 1 shows a typical IRT formulation. Eq. 1 shows a typical item response theory.

$$P(y = 1|u, t) = c_t + (1 - c_t)\sigma\left(a_t\left(\theta_u - d_t\right)\right) \tag{1}$$

Here, σ is a logistic sigmoid function, which is widely used in probabilistic models because, for a value $x \in \mathcal{R}$, $0 < \sigma(x) < 1$ holds.

$$\sigma(x) = \frac{1}{(1 + \exp(-x))} \tag{2}$$

Eq. 1 has four parameters to estimate, as listed and explained below.

θ_u. Ability parameter of the user u.

d_t. Difficulty parameter of the item t.
a_t. Discrimination parameter of the item t.
c_t. Guessing parameter of the item t

The ability of every user $u \in U$ is modeled by user parameter θ_u. The higher θ_u is, the greater is the ability that our system regards user u as having.

The difficulty of each item $t \in T$ is modeled by d_t. The higher d_t is, the more difficult item t is and the fewer users can answer correctly.

The latter parameters are optional. Discrimination parameter a_t determines how steep the item characteristic curve of item t is. The higher a_t is, the steeper the item characteristic curve is.

Guessing parameter c_t determines the probability of test takers guessing the correct answer. This parameter is typically used to model multiple choice items. For example, if a test taker is supposed to choose one item from five choices, then $c_t = \frac{1.0}{5.0} = 0.20$.

4.1.2. Rasch Model. Our system simply assumes items to be words.

$$P(y = 1|u, t) = \sigma\left(\theta_u - d_t\right) \tag{3}$$

By comparing Eq. 1 with Eq. 3, you can find that Eq. 3 is a special case of Eq. 1. Indeed, Eq. 3 is derived from Eq. 1 by substituting $a_t = 1$ and $c_t = 0$ for $\forall t \in T$, where $a_t = 1$ means that all the questions are regarded as equally discriminative and $c_t = 0$ means that the model assumes that it is impossible to guess the correct answer.

4.2. Logistic Regression

This section introduces logistic regression, which is a kind of "supervised classification" in machine learning literature [Bishop 2006].

In supervised classification, a "classifier" is first trained with "training data". A training datum can be represented as $(y, \boldsymbol{x})$, which is a pair of "input vector" $\boldsymbol{x}$ and "class" $y \in \mathcal{Y}$, where $\mathcal{Y}$ is the "domain" of the classes and $\mathcal{X}$ is the domain of input vectors. A classifier has a "parameter" vector, $\boldsymbol{w}$, that stores all the information for the current status of the training, where $\boldsymbol{w}$ determines the classifier. Given another input vector $\boldsymbol{x}$ and parameter vector $\boldsymbol{w}$, the classifier further predicts the class of the input vector. When it is predicting, the classifier does not use the input vector $\boldsymbol{x}$ as-is: rather, it first constructs a feature vector $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{\phi}(u, t)$.

Supervised classification methods can be categorized into two types: probabilistic or non-probabilistic. Probabilistic methods calculate the probability that y belongs to a certain class given input vector $\boldsymbol{x}$. They then classify $\boldsymbol{x}$ as the most probable class in domain $\mathcal{Y}$. Logistic regression is one of the most typical of these methods. Non-probabilistic methods such as Support Vector Machines (SVM) do not use probability explicitly but simply maps $\boldsymbol{x}$ to some $y \in Y$. For our system, probabilistic classification is preferable because, even if the prediction fails, it tells us how significant the failure is in the form of a probability. This is beneficial in this application because it enables users to prioritize the predictions to correct.

Given a training datum $(y, \boldsymbol{x})$, logistic regression directly models the conditional probability of y given $\boldsymbol{x} \in \mathcal{X}$ by a sigmoid function. In this application, input vector $\boldsymbol{x}$ consists of two components: the index of user $u \in \{1, \ldots, |U|\}$ and the index of word $t \in \{1, \ldots, |T|\}$. Thus, $\boldsymbol{x} = (u, t)$.

Random variable y represents how well user u knows word t. In this study, we simply assumed that this is represented by two states: "know" and "don't know". That is, $y = 1$ when user u knows word t and $y = 0$ otherwise. Thus, the domain of y is $\mathcal{Y} = \{1, 0\}$ in this application. Limiting y to be a binary random variable has a benefit: it reduces the number of variables to optimize. In our system, we need to optimize vector $\boldsymbol{w}$ so that it fits the data. If y can take more than two classes, say K classes, then $(K-1)|\boldsymbol{w}|$ variables need to be optimized. However, if y is binary, then optimizing $|\boldsymbol{w}|$ variable is sufficient; the probability of one class, $y = 1$ in Eq. 4, also determines the probability of the other class $y = 0$, as shown by Eq. 4.

$$\begin{array}{rcl} P(y=1|\boldsymbol{\phi};\boldsymbol{w}) & = & \sigma\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(u,t)\right) \\ P(y=0|\boldsymbol{\phi};\boldsymbol{w}) & = & 1-\sigma\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(u,t)\right) \\ & = & \left(1-\sigma\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(u,t)\right)\right)^{1-y} \end{array} \tag{4}$$

First, we explain how the system, given a weight vector $\boldsymbol{w}^{(k)}$, constructs binary classifier h to be represented as Eq. 5, in which the notation in Eq. 4 is used. Here, k is the index of updates, i.e., the number of times the weight vector has been changed.

$$h(u,t;\boldsymbol{w}^{(k)}) = \log\frac{P(y=1|\boldsymbol{\phi}(u,t);\boldsymbol{w}^{(k)})}{P(y=0|\boldsymbol{\phi}(u,t);\boldsymbol{w}^{(k)})} \tag{5}$$

Second, we explain how the system learns the weight vector $\boldsymbol{w}^{(k)}$ from the accumulated log which stores all the clicks of all the users indexed by $n \in \{1,\ldots,N\}$. In Eq. 6, the notation introduced in Eq. 4 also combines $P(y=1|\boldsymbol{\phi};\boldsymbol{w})$ and $P(y=0|\boldsymbol{\phi};\boldsymbol{w})$ into one representation $P(y|\boldsymbol{\phi};\boldsymbol{w})$.

$$P(y|\boldsymbol{\phi};\boldsymbol{w}) = \sigma\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}\right)^{y}\left(1-\sigma\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(u,t)\right)\right)^{1-y} \tag{6}$$

Thus, the negative log likelihood $E(\boldsymbol{w})$ of Eq. 6 is derived as shown in Eq. 7. The maximum likelihood estimator of Eq. 6 is given by minimizing Eq. 7.

$$\begin{array}{rcl} E(\boldsymbol{w}) & = & -\log\left(\prod_{n=1}^{N} P(y_n|\boldsymbol{\phi}_n;\boldsymbol{w})\right) \\ & = & -\sum_{n=1}^{N}\log\left(P(y_n|\boldsymbol{\phi}_n;\boldsymbol{w})\right) \end{array} \tag{7}$$

4.3. Relation to Rasch Model

This section shows that IRT is a special case of logistic regression when parameter vector $\boldsymbol{w}$ and feature vector $\boldsymbol{\phi}$ are modeled as follows. Let $|A|$ denote the size of set A. Let $\oplus$ denote the direct sum of two vectors; i.e., given two vectors

$$\begin{array}{rcl} \boldsymbol{r} & = & (r_1,\ldots,r_{N_r}) \in \mathbf{R}^{N_r} \\ \boldsymbol{s} & = & (s_1,\ldots,s_{N_s}) \in \mathbf{R}^{N_s} \end{array} \tag{8}$$

Eq. 9 shows $\boldsymbol{r} \oplus \boldsymbol{s}$.

$$\boldsymbol{r} \oplus \boldsymbol{s} = (r_1,\ldots,r_{N_r},s_1,\ldots,s_{N_s}) \in \mathbf{R}^{N_r+N_s} \tag{9}$$

In addition, this paper uses the following notation: a vector representing user language ability,

$$\boldsymbol{\theta} = (\theta_1,\ldots,\theta_u,\ldots,\theta_{|U|}) \tag{10}$$

and a vector of the difficulty of the words,

$$\boldsymbol{d} = (-d_1,\ldots,-d_t,\ldots,-d_{|T|}) \tag{11}$$

The Rasch model is a special case of logistic regression where $\boldsymbol{w}$ and $\boldsymbol{\phi}(u,t)$ in Eq. 4 are defined as in Eq. 12. $\boldsymbol{e}_u$ and $\boldsymbol{e}_t$ are unit vectors: $\boldsymbol{e}_u \in \mathbf{R}^{|U|}$, which denotes a unit vector whose u'th element is 1 while all the other elements are 0. $\boldsymbol{e}_t \in \mathbf{R}^{|T|}$, which denotes a unit vector whose t'th element is 1 while all the other elements are 0.

$$\begin{array}{rcl} \boldsymbol{w} & = & \boldsymbol{\theta} \oplus \boldsymbol{d} \\ \boldsymbol{\phi}(u,t) & = & \boldsymbol{e}_u \oplus \boldsymbol{e}_t \end{array} \tag{12}$$

Eq. 12 can be extended by adding additional features, as shown in Eq. 13.

$$\begin{array}{rcl} \boldsymbol{w} & = & \boldsymbol{\theta} \oplus \boldsymbol{d} \oplus \boldsymbol{w}_a \\ \boldsymbol{\phi}(u,t) & = & \boldsymbol{e}_u \oplus \boldsymbol{e}_t \oplus \boldsymbol{\phi}_a(u,t) \end{array} \tag{13}$$

Table I. Parameter estimation methods for logistic regression.

	Optimal	Batch/Online
L-BFGS [Liu and Nocedal 1989]	$\surd$	Batch
Trust Region [Lin et al. 2008]	$\surd$	Batch
SGD [Bishop 2006]		Online

Here, $\boldsymbol{\phi}_a$ is an additional feature vector that depends on u and t, and $\boldsymbol{w}_a$ is the weight vector for $\boldsymbol{\phi}_a(u, t)$. As the additional feature vector provides more information about u and t, when selected appropriately, it improves the system's prediction performance.

Our system uses $\boldsymbol{\phi}_a$ to introduce word difficulty. An appropriate choice of $\boldsymbol{\phi}_a$ improves accuracy. Remember that our system needs to estimate user language ability $\boldsymbol{\theta}$ and word difficulty $\boldsymbol{d}$. Among these two, not much information about language ability is expected to be available ahead of time because many users are anonymous by the nature of the Web community. In contrast, much is known about word difficulty $\boldsymbol{d}$. While many measures of word difficulty have been proposed, most of them are based on word frequency in a large corpus because word frequency has been demonstrated to be good measure of word difficulty for ESL (English as a second language) learners [Tamayo 1987]. Thus, we used two kinds of features for word difficulty: "Google 1-gram" and "SVL". Google 1-gram is created from the raw word frequency taken from [Brants and Franz 2006]. Taken from approximately a trillion Web pages, [Brants and Franz 2006] is one of the largest word frequency lists known. Specifically, we used the value of normalized $-\log(\frac{f_t}{\sum_t f_t})$, where f_t is the word frequency of word t in [Brants and Franz 2006]. Although also based on word frequency, SVL [SPACE ALC Inc. 1998] is a word difficulty measure manually checked by English native speakers for Japanese ESL learners. SVL12000 covers 12,000 words and has 12 levels.

A Rasch model is not simply applicable to this system because the input data are too sparse. However, by using the notation used for logistic regression, we can tackle this problem by just adding more features to $\boldsymbol{\phi}$. In particular, we can approximate average word difficulty by using word frequencies; there are many report that word frequency can be used as a measure of word difficulty [Tamayo 1987]. We used word frequencies from the Google corpus [Brants and Franz 2006], whose n-gram counts are claimed to have been "generated from approximately 1 trillion word tokens of text from publicly accessible Web pages".

5. PARAMETER ESTIMATION

This section introduces parameter estimation methods for logistic regression. The main parameter estimation algorithms are shown in Table I. To train the classifier for our system, online learning methods are preferable to batch learning methods because the click logs of our system are accumulated in an online manner. Batch learning may, however, be applicable to our system, for example, for training the classifier periodically.

The "Optimal" column shows whether the algorithm can find the global optimal solution of the objective function of a logistic regression. The "Batch/online" column shows whether the algorithm is a batch algorithm or an online algorithm.

L-BFGS [Liu and Nocedal 1989] is a widely used method of estimating logistic regression parameters. It is a variant of quasi-Newton method. It consumes a memory space only linear to the number of features while a naïve method consumes a memory space squared to the number of features.

Trust region [Lin et al. 2008] is another recently proposed batch method for optimizing binary logistic regression parameters. It can optimize the parameters faster than L-BFGS, but it is limited to logistic regression whereas L-BFGS is more general and more widely applicable. An implementation of Trust region is given in [Fan et al. 2008] and this implementation was used for the evaluation later described.

5.1. Stochastic Gradient Descent (SGD)

The proposed system uses SGD (Stochastic Gradient Descent) to estimate the parameters because it is an online algorithm. While a batch algorithm requires the whole training data to be given from

Table II. Dale's scale.

Stage 1	"I never saw it before."
Stage 2	"I've heard of it, but I don't know what it means."
Stage 3	"I recognize it in context - it has something to do with ..."
Stage 4	"I know it."

the beginning to the end, an online algorithm can process every record of the data one by one. An online algorithm suits for a Web application, because users of a Web application do not feed the whole data at a time.

SGD is derived from steepest gradient descent, a batch algorithm, by removing summation over the data points. The steepest gradient descent for logistic regression is shown in Eq. 14 where $\nabla E_n(\boldsymbol{w}) = \left(y_n - \sigma\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}_n\right)\right)\boldsymbol{\phi}_n$. Here, $\boldsymbol{\phi}_n = \boldsymbol{\phi}(\boldsymbol{x}_n)$, where $\boldsymbol{x}_n$ is the $n \in \{1, \dots, N\}$th click log among all N click logs.

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} - \eta \sum_{n=1}^{N} \nabla E_n(\boldsymbol{w}^{(k)}) \tag{14}$$

Eq. 14 contains the summation of E_n over all the data, so it is a batch algorithm. In this case, an online algorithm is easily derived by removing the summation over n from Eq. 14. The resulting online algorithm is SGD, the algorithm used in our system. SGD is given by

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \eta_k \nabla E_n(\mathbf{w}^{(k)}) \tag{15}$$

In Eq. 15, η_k is defined as $\eta_k = \frac{1}{\lambda(k+k_0)}$, where λ, k_0 are the parameters of SGD and η_k is defined in this way so that Eq. 15 converges [Novikoff 1963]. Note that η is usually a constant in Eq. 14. $Update(\boldsymbol{w}^{(k)}, u, t, y)$ in Figure 3 is defined by Eq. 15.

6. EXPERIMENTS

This section presents the results of our experiments and discusses them.

6.1. Evaluation Data

We developed a database of the vocabulary knowledge of 16 human subjects for 12,000 words for training and evaluating our system. This database is a matrix of the numbers representing the degree of vocabulary knowledge, where the rows correspond to the human subjects and the columns correspond to the words. The vocabulary knowledge was obtained by assessing the *language ability of humans* introduced in [Read 2000].

The database was developed by firstly determining the set of words for assessing the language ability of humans. The word set was selected from the Standard Vocabulary List 12000 (SVL12000) [SPACE ALC Inc. 1998]. SVL12000 lists the most fundamental 12,000 words that an English learner should learn and they have been checked by native English speakers. Then, the questions for assessing the subject's language ability were developed. Each human subject answered the question for every word in SVL12000.

In [Read 2000], two methods are introduced to make questions for measuring the degree of human language ability.

Multiple-choice items. A question consists of a word and multiple choices. The subjects are told to choose the choice that has the same meaning as the word.

Self-report scale. A subject answers with freely chosen words indicating how well he/she knows the word in his/her subjective judgment.

The Self-report scale was used for developing the evaluation data because creating and answering multiple-choice items for 12,000 words has a much higher cost burden than the Self-report scale.

There have been several studies on designing scales for the Self-report scale. The scale shown in Table II was proposed in early research by Dale [Dale 1965]. Paribakht & Wesche [Paribakht and Wesche 1997] proposed the scale shown in Table III. Although the first four ranks on the scales

Table III. Paribakht & Wesche's elicitation scale.

I	"I don't remember having seen this word before."
II	"I have seen this word before, but I don't know what it means."
III	"I have seen this word before, and I think it means (synonym or translation)."
IV	"I know this word. It means (synonym or translation)."
V	"I can use this word in a sentence: (Write a sentence.) (If you do this section, please also do Section IV)

Table IV. Our scale for evaluation.

1	never seen the word before
2.1	probably seen the word before
2.2	absolutely seen the word before but don't know its meaning / tried to learn the word before but forgot its meaning
3	probably know the word's meaning / able to guess the word's meaning
4	absolutely know the word's meaning

show some correspondence between Dale's scale and Paribakht & Wesche's scale, the latter has one more additional rank, rank V, which tests the subject's writing ability by testing whether the subject can use the word correctly in a sentence. Rank V is additional to the other ranks; those who do rank V must also do rank IV, as described in the note in Table III. We created the scale shown in Table IV. It is based on both Dale's scale and Paribakht & Wesche's scale with two major modifications. The first modification is that we omitted rank V because that exists for testing writing ability rather than testing reading ability, which our system tries to support. The second modification is that knowledge of synonyms and translations is omitted in our ranks. The reason for this is to prevent bias in the results. As it takes time for the subjects to give a synonym or translation, they may hesitate to choose an item that requires one, especially if the test covers many words. Another modification to Dale's scale is that we divided this Rank 2 into two ranks: 2.1 and 2.2. This modification was introduced because we wanted to discriminate the case where the subject has tried to learn the word before from the case where the subject simply has seen the word before. Dale's scale does not discriminate these two because it was originally for testing L1 language ability, and words are usually learned incidentally in L1 acquisition.

16 students mostly from graduate schools of the University of Tokyo participated in the test as human subjects. Each student answered 12,000 questions corresponding to the 12,000 words using our scale (Table IV). Note that the ranks in Table IV are numbered to correspond to those of Dale's scale (Table II) and Paribakht & Wesche's scale (Table III). The subjects saw a simple ordinal numbering $1, 2, 3, 4, 5$, so that numbering did not affect their results.

The scale in Table IV needs to be binarized to evaluate our system because our system eventually classifies words into "known" and "unknown" to a user. This scale binarization was performed as follows: Only rank 4 in Table IV was regarded as $y = 1$, i.e., the case where user u knows word t. All the other ranks were regarded as $y = 0$, i.e., the case where user u does not know word t.

The aim of this binarization is as follows: as the goal of our system is to support reading by automatically showing glosses of words, in our system's evaluation, the criteria when binarizing these scales should be whether or not the user needs the glosses of the words. Those words in ranks 1, 2.1, and 2.2 clearly need to be glossed when they appear in texts because the subject does not know them, so they surely hamper reading. The words in rank 3 also need to be glossed because, although the user may be able to "guess" their meanings, these guesses could be wrong and, without glosses being shown when the words appear in a text, the user cannot be informed of his/her wrong guesses and might not fully comprehend the text containing the words or might misinterpret the text, either of which would hamper reading.

Table V. Values of accuracy (%) for IRT and LR.

	$N = 10$	30	100	300	600
IRT	68.33	73.69	74.65	74.65	74.60
LR	**73.25**	**77.89**	**79.09**	**80.03**	**80.01**

6.2. Evaluation by Accuracy

The database was divided into a training set, development set, and test set. 600 words were randomly chosen for the training set, 1400 words were randomly chosen for the development set, and 9999 words were randomly chosen for the test set. Accuracy was defined as the percentage of the words that our system correctly answered among the 9999 words in the test set.

The evaluation setting simulated the case where a new user starts using our system with a specified log. Every user among the 16 subjects was assumed to start using the system as a new user and to click at most 600 words. This number 600 was chosen so that everyone could click them within about 5 minutes.

The system was trained with data from **smart.fm** [Cerego Japan Inc. 2009] instead of our system's log because no log had been accumulated at the start. **smart.fm** is an implementation of a computer-assisted vocabulary acquisition (CAVOCA) system in the field of computer-assisted language learning (CALL). The purpose of a CAVOCA system is to support its users, typically ESL learners, so that they can learn as many English words as possible in a fixed period. A CAVOCA system works as follows: First, the CAVOCA system gives the user a word list and the user selects words in the list to learn. If the user finds words that he/she already knows, he/she can check them and skip them. For our system, we used the checked words as training data because they are regarded as known words and the unchecked words in the word list are regarded as unknown words. Note that words absent from the word list were not used. Once the set of words to learn has been determined, the CAVOCA system automatically generates a set of questions about the words and presents these questions to the user. The CAVOCA system continues generating questions and the user continues answering them until he/she is able to answer the whole set of questions correctly. The question types include multiple-choice questions and spelling questions. While we obtained the records for 10,526 **smart.fm** users, we eventually chose 675 users from amongst these and used their records because many users simply tried **smart.fm** and did not use it seriously and repeatedly. These 675 users were chosen according to the following criteria: selection of at least 100 words for learning and 15% or more skipped words out of the total number of words in the word lists.

Our system was trained with data created from **smart.fm** (N_0) and at most 600 new user words (N_1) as training data. Here, $N = N_0 + N_1$, where N denotes the same N as in Section 5.1. The hyperparameters were tuned for the development set, and the accuracy of our system was measured on the test set. The accuracy was defined as the average percentage for the case where the classifier predicted correctly over the test sets of the 16 subjects.

The Rasch model is a kind of IRT that can be defined as a logistic regression by providing user-IDs and word-IDs as features in logistic regression. Our model is an extension of the Rasch model with the addition of word difficulty features in logistic regression. This section describes how we evaluated the effectiveness of the word difficulty features. [Fan et al. 2008] was used for both IRT and LR with L2-regularization. The regularization parameter was selected from $1.0, 0.5, 2.0$ and the value that gave the highest accuracy in the development set was used in the test. Note that since IRT is a logistic regression that does not use word difficulty features, evaluating the case where no word difficulty features are used corresponds to evaluating the effectiveness of IRT.

The results are shown in Table V and Figure 5. Both show that "LR", the case that includes word difficulty features, exhibited considerably higher accuracy than "IRT". This means that word difficulty features are effective because the difference between "LR" and "IRT" lies only in the word difficulty features. Thus, the use of word difficulty features is confirmed to be reasonable.

Another observation from Figure 5 is that the accuracy of both settings seemed to saturate in the range where the number of training data was over 300. "LR" achieved about 5% higher accuracy than "IRT" as a result of using word difficulty features.

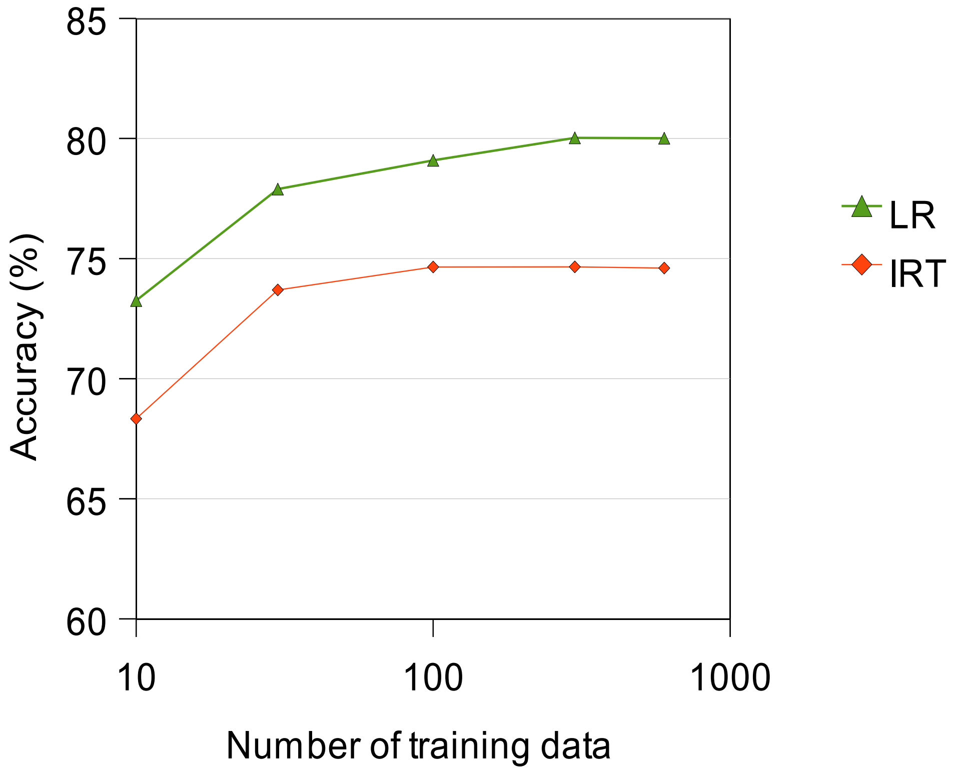


Fig. 5. Comparison of logistic regression and IRT.

6.3. Comparing Logistic Regression with Other Classifiers

There are many algorithms for binary classification besides logistic regression. Our system can be trained with them by using the same features. This section compares logistic regression with other binary classification algorithms in terms of accuracy. The following algorithms were compared with logistic regression.

(1) SVM (Linear)
(2) SVM (RBF)

"SVM (Linear)" is a support vector machine with a linear kernel. This algorithm was chosen for the comparison because a support vector machine is a state-of-the-art algorithm for binary classification. Unlike a support vector machine with a Gaussian kernel, a support vector machine with a linear kernel cannot fully classify nonlinear data. However, this does not cause much problem with high-dimensional data because there are enough dimensions to classify the given data linearly. Explanations of support vector machines are found in [Cristianini and Shawe-Taylor 2000]. Again, [Fan et al. 2008] was used for the implementation. The performance of a support vector machine is known to be affected by the value of regularization parameter C. In this evaluation, C was chosen from $1.0, 2.0, 0.5$. The value was tuned for the development set.

"SVM (RBF)" is a support vector machine with a Gaussian kernel, so it is another state-of-the-art algorithm for binary classification. [Fan et al. 2008] was used for the implementation. Unlike all the

Table VI. Prediction accuracy (%) for various algorithms.

	$N_1 = 10$	30	100	300	600
SVM (Linear)	74.78	**78.08**	78.88	79.20	79.27
SVM (RBF)	67.61	77.27	**79.16**	79.55	79.91
SGD	**76.95**	77.94	78.52	78.46	78.39
LR	73.25	77.89	79.09	**80.03**	**80.01**

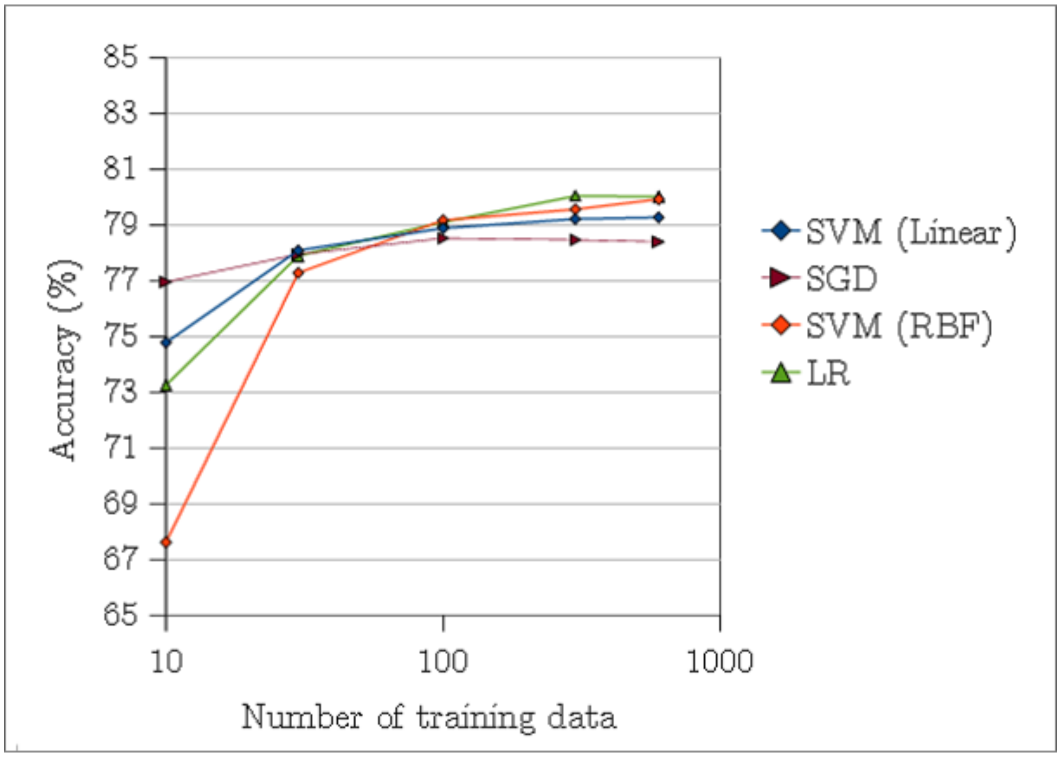


Fig. 6. Comparison of logistic regression and other algorithms.

other algorithms, "SVM (RBF)" takes hours to optimize. The value of regularization parameter C was set to 1.0.

"Stochastic Gradient Descent" (SGD) is the SGD described so far. Note that no regularization is performed in SGD.

Finally, "LR" denotes the results for logistic regression. Again, LR was optimized with L2-regularization using [Fan et al. 2008]. The regularization parameter was selected from $1.0, 0.5, 2.0$, and the value that gave the highest accuracy in the development set was used for the test.

Accuracy values for logistic regression and other algorithms are listed in Table VI. Logistic regression achieved the highest accuracy when the number of the training data was 300 and 600. It also achieved nearly the highest accuracy for $30, 100$. This means that logistic regression is sufficiently accurate compared with the other algorithms. Thus, the use of logistic regression is reasonable for this application.

When the numbers of the training data were 10 and 30, SGD outperformed LR and achieved the highest and the second highest accuracy respectively. This result is encouraging because it suggests that SGD is quick to adapt to a user. SVM (RBF) had the lowest accuracy when the number of training data was set to 10. We attributed this to the amount of training data being too small and to parameter C of the SVM (RBF) being fixed to 1.

As shown in Figure 6, all the algorithms saturated in the range where the number of training data exceeded 300 and even the highest of accuracy was only about 80%. This shows that about 80% is the accuracy limit owing to the nature of the given data.

6.4. Simulation of Reading

This section evaluates our system by simulating users' document-reading processes using the data mentioned in 6.1. We simulated the case where each user reads all the documents in the "Brown corpus", which consists of 500 documents taken from various sources for balance. The average number of words in a Brown Corpus document is 2029.

Leave-one-out evaluation was performed. First, we selected one user for the development to tune the parameters of the classifiers. The data from this user were used for development purposes only, and never used for training or testing. In each fold, a test user was selected, and all the data of the other users were used as training data of "training 1", i.e., to learn the word difficulty, as explained later.

1 The next document to read was given randomly.
2 Given the document to read and a test user, each classifier classified and glossed all the words in the document into known or unfamiliar.
3 Using the vocabulary knowledge collected in 6.1, we simulated the test user reading the documents and correcting all the words to reflect his or her knowledge of those words.
4 Each classifier was trained from these corrections. Go back to 1.

The evaluation of our system involved both its ability to estimate difficulty of words and its ability to estimate the users' language abilities. As the focus of this evaluation was the latter, we wanted to make the training data separate. Therefore, the training data were divided into two parts: "training 1" and "training 2". Here, "training 1" corresponds to the logs accumulated so far and was the part for difficulty of words.

The following three methods were compared. The C hyperparameter for the logistic regression was chosen from $100.0, 10.0, 1.0, 0.1, 0.01, 0.001$ and the value that achieved the highest accuracy (in number of types) using the data from the development user was selected. For hyperparameters of SGD, the setting $\lambda = 1.0, k_0 = 0$ was used.

Unfamiliar. The baseline. This simulates the case where no prediction is available; i.e., the user clicks all unknown words.

LR. simulates the case where in both "training 1" and "training 2", the costly batch learning of a logistic regression, is used.

LR+SGD. simulates the case where the combined method is used. That is, in "training 1", a costly batch learning of a logistic regression is used. Then, the resulting weight vector (i.e., parameter vector) is used as an initial weight vector of SGD, an efficient online learning of logistic regression to tune only the weight for the test user's language ability parameter. All the other parameters including the parameters for difficulty of words are fixed during "training 2".

First, we evaluated our system in terms of the number of clicks required. Table VII lists the average percentage of words to be clicked, i.e., the total number of clicks divided by the total number of words in the 500 documents. As the denominator is a constant, these percentage values are proportional to the required number of clicks. Note that, here, the number of words means the number of occurrences of words, not the number of types of words as used in the previous section. The users were sorted by the percentage of their unfamiliar words in descending order. The asterisks are T-test trial results for each method versus "Unfamiliar", the baseline. "*" denotes that the p-value is lower than 0.05, "**" denotes that the p-value is lower than 0.01, and "***" denotes that the p-value is lower than 0.001.

Investigating Table VII, we can find that, from user0 to user7, "LR" significantly reduced the number of clicks and from user0 to user6, "LR+SGD" significantly reduced the number of clicks. Intuitively, the smaller the number of the user's unfamiliar words becomes, the more difficult for the

Table VII. Average percentage of words to be clicked in a document in the corpus consisting of 500 documents.

User	Unfamiliar (%)	LR	LR+SGD
user0	64.74	**25.51** ***	29.89 ***
user1	21.21	**12.36** ***	12.80 ***
user2	12.55	**10.25** ***	11.58 ***
user3	11.89	**6.93** ***	7.62 ***
user4	9.82	**7.09** ***	7.51 ***
user5	7.33	**5.65** ***	6.94 *
user6	5.33	**4.03** ***	4.98 *
user7	5.32	**4.55** ***	5.15
user8	**5.10**	5.52 ***	6.68 ***
user9	4.56	**4.53**	5.77 ***
user10	4.34	**4.02** ***	5.07 ***
user11	**1.80**	2.42 ***	3.26 ***
user12	**1.51**	2.08 ***	2.79 ***
user13	**1.08**	2.05 ***	2.56 ***
user14	**0.39**	2.06 ***	2.84 ***

Table VIII. Average percentages of remaining unfamiliar words in the 500 documents.

User ID	Unfamiliar (%)	LR	LR+SGD
user0	64.74	**14.52** ***	14.66 ***
user1	21.21	7.73 ***	**6.07** ***
user2	12.55	6.28 ***	**4.82** ***
user3	11.89	4.59 ***	**3.26** ***
user4	9.82	4.26 ***	**3.25** ***
user5	7.33	3.34 ***	**2.56** ***
user6	5.33	2.30 ***	**1.71** ***
user7	5.32	2.54 ***	**1.73** ***
user8	5.10	2.91 ***	**2.38** ***
user9	4.56	2.42 ***	**1.93** ***
user10	4.34	2.13 ***	**1.66** ***
user11	1.80	0.76 ***	**0.57** ***
user12	1.51	0.54 ***	**0.43** ***
user13	1.08	0.38 ***	**0.32** ***
user14	0.39	0.17 ***	**0.13** ***

system to make significant reductions in the number of clicks. However, this case is not problematic for the user because, if the user has a large enough vocabulary to read the documents without any support, the user does not need reading support in the first place.

Readers do not always need to be familiar with all the words in a document to read that document. [Nation 2006] surveyed and investigated the percentage of words necessary to read documents and reported that one must know from 95% to 98% of the words in a running text to read documents sufficiently without assistance.

We also evaluated our system in terms of the percentage of "remaining unfamiliar words", as shown in Table VIII. As the words predicted to be unknown to a user were glossed in advance, the users could know the meaning of those glossed words when using our system. Thus, the unfamiliar words remaining were the words predicted to be known to a user, though the user actually did not know them. We call these words "remaining unfamiliar words". The meaning of asterisks is the same as in the previous table.

From Table VIII, we can see that the numbers of remaining unfamiliar words were significantly decreased for all users. In particular, for user2–to user8, the system significantly made unreadable documents nearly readable as the average remaining unfamiliar became lower than 5% while originally being above 5%, according to [Nation 2006]. Moreover, interestingly, the proposed "LR+SGD" achieved better results than the costly "LR" method. This is probably due to the fact that in “LR+SGD”, SGD updates only the language ability parameters while “LR” tries to adjust all the parameters and causes overtraining.

Finally, we compared the time required to train the logistic regression in “LR” and SGD in “SGD”. While the total time to finish the training for 500 documents was 0.43 (s) in SGD on average, LR took 1975.53 (s) and the t-test was of course significant as the p-value was under 10^{-22}. We can easily see that “LR+SGD” was the most efficient.

7. CONCLUSIONS

We described a new glossing system that can predict words unknown to the user by utilizing logs that contain valuable information about a user’s vocabulary. Although existing glossing systems are helpful for English learners because they provide the meaning of a word by displaying it in a pop-up window when the user encounters an unknown word and clicks on or mouses over it, they waste this log information.

We investigated models for our system’s prediction. The use of logistic regression was found to be appropriate because the Rasch model, a simple form of item response theory (IRT) widely used to assess human language ability, is also a logistic regression that uses a specific feature vector. We extended IRT by introducing word difficulty features and achieved accuracy higher than that of straightforward IRT.

We also proposed a method of training parameters suitable for our system. IRT and the extended IRT can estimate both the difficulty of words and the users’ language abilities simultaneously from the accumulated logs as one parameter vector. Note that the estimated difficulty of the words is not simple usage of existing word difficulty measures, but is adjusted using the combination of existing word difficulty measures to fit the training data from the system users. For example, if an existing word difficulty measure is not correlated with the system users, the weight word difficulty measure approaches zero and is nearly omitted.

Our system requires two kinds of estimation: one is estimation of the difficulty of the words so that it suits the users of our system; the other is estimation of user language levels including those of new users. The former is more stable because the difficulty of a word does not change suddenly, while the latter is more dynamic because user language levels differ from user to user and our system needs to become accustomed to a new user quickly.

To meet these requirements, the proposed estimation method combines both batch learning and online learning by sharing the parameter vector. The costly batch learning is used to learn the difficulty of words while our system is idle, and the estimated parameter is copied to the stochastic gradient descent (SGD) learner. SGD updates only the users’ language ability parameters quickly in an online manner and is thus appropriate for our system where the logs are accumulated in an online manner.

We evaluated our system in terms of the number of clicks and the percentage of unfamiliar words remaining even after the prediction and glossing. To evaluate our system, we collected knowledge about 12,000 words from 16 participants and used these data to simulate users’ reading of the Brown corpus.

The simulation results show that our system can significantly reduce the number of clicks for users who know up to 94.7% of the running words in a document (a.k.a., coverage). As [Nation 2006] shows that a reader should have 95% or higher coverage of a document to read it sufficiently, this shows that our system can reduce the number of clicks for most readers with insufficient word coverage to read documents. The simulation results also show our system can significantly reduce the number of remaining unfamiliar words after the prediction and glossing for all users.

Finally, we compared the time required to train the costly batch model and our combined method. As the latter took 0.43 s for the training of 500 documents on average while the former took 1975.53 s, the effectiveness of the combined method is shown.

To conclude, the combined use of batch learning for mainly learning word difficulty and online learning for mainly learning the users’ language ability is efficient for increasing the training speed, decreasing the required number of clicks, and decreasing the number of unfamiliar words.

ACKNOWLEDGMENTS

The authors express deep appreciation for the co-operation of the experiment participants.

REFERENCES

BAKER, F. AND KIM, S. 2004. *Item response theory: Parameter estimation techniques*. CRC.

BISHOP, C. M. 2006. *Pattern recognition and machine learning*. Springer.

BRANTS, T. AND FRANZ, A. 2006. *Web 1T 5-gram Version 1*. Linguistic Data Consortium, Philadelphia.

CEREGO JAPAN INC. 2009. smart.fm. System available at `http://smart.fm/`.

COOLEST.COM INC. 2002. popjisyo.com. `http://www.popjisyo.com/`.

CRISTIANINI, N. AND SHAWE-TAYLOR, J. 2000. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge Univ Pr.

DALE, E. 1965. Vocabulary measurement: techniques and major findings. *Elementary English 42*, 895–901, 948.

FAN, R., CHANG, K., HSIEH, C., WANG, X., AND LIN, C. 2008. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research 9*, 1871–1874.

LIN, C., WENG, R., AND KEERTHI, S. 2008. Trust region Newton method for logistic regression. *The Journal of Machine Learning Research 9*, 627–650.

LIU, D. AND NOCEDAL, J. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming 45*, 1, 503–528.

NATION, I. S. P. 2006. How large a vocabulary is needed for reading and listening? *Canadian Modern Language Review 63*, 1, 59–82.

NORI. 2005. Firedictionary.com. `http://www.firedictionary.com/`.

NOVIKOFF, A. B. 1963. On convergence proofs for perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*. Vol. 12. 615–622.

PARIBAKHT, T. AND WESCHE, M. 1997. Vocabulary enhancement activities and reading for meaning in second language vocabulary acquisition. *Second language vocabulary acquisition: A rationale for pedagogy*, 174–200.

POPIN INC. 2008. popin. `http://www.popin.cc/en/home.html`.

READ, J. 2000. *Assessing Vocabulary*. Cambridge University Press.

SPACE ALC INC. 1998. Standard vocabulary list 12,000. Data available at `http://www.alc.co.jp/goi/PW_top_all.htm`.

SUMITA, E., SUGAYA, F., AND YAMAMOTO, S. 2005. Measuring non-native speakers' proficiency of English by using a test with automatically-generated fill-in-the-blank questions. In *Proceedings of the Second Workshop on Building Educational Applications Using NLP*. Association for Computational Linguistics, Ann Arbor, Michigan, 61–68.

T. D. RUDICK. 2001. Rikai. `http://www.rikai.com/`.

TAMAYO, J. M. 1987. Frequency of use as a measure of word difficulty in bilingual vocabulary test construction and translation. *Educational and Psychological Measurement 47*, 4, 893–902.

Received October 2010; revised January 2011; accepted April 2011