Vol.2010-HCI-138 No.4
2010/5/14

en.newikipedia.org：英語版 Wikipedia 中のユーザが知らない英単語を予測するユーザ参加型読解支援システム

江原 遥[†1] 二宮 崇[†2]
清水伸幸[†2] 中川裕志[†2]

筆者らは，任意の英文 Web ページに対して，クリックログの解析を基にユーザ知らない英単語をユーザの英語力に応じて予測することで，Web ページの読み込み時にそれらの語に対して訳を予め付与する，ユーザ参加型の読解支援システム SocialDict を提案している．しかし，SocialDict では，ログインや URL の入力がユーザに取って手間となりユーザビリティが低下する問題，対象とする Web ページに対する深い解析を必要とする特徴量を予測に利用することが難しいという問題があった．本稿では，任意の英文 Web ページを対象とすることを犠牲にする代わりに，Wikipedia に特化したシステム NEWikipedia を提案する．これにより，ログインや URL の入力を不要としてユーザビリティを向上させ，LDA を用いて得られたトピックごとの単語頻度の特徴量を判別に利用することが可能となる．

A reading support system for English Wikipedia with prediction capability of the words unknown to the users

Yo Ehara,[†1] Takashi Ninomiya ,[†2]
Nobuyuki Shimizu [†2] and Hiroshi Nakagawa[†2]

We have been proposing a system to support reading English Web pages for ESL (English as a second language) users by annotate the words they may not know by personalized prediction of those words using the word click logs. However, after months of running our system, we found several problems: first, while our system requires user to login and specify the URL of the Web page to annotate, these actions themselves are cumbersome to the users , and thus, decrease usability. Second, while our system is targeting all English Web pages, this wide range of target Web pages prevents use of features that require deep analysis of the target Web pages in advance for the prediction. In this paper, we propose a possible solution to this problem by sacrificing targeting wide range of Web pages and specialize the system to Wikipedia.

1. はじめに

近年，英文 Web ページを読むニーズが増えているのに伴い，読解支援の重要性が増している．第二言語で書かれた Web ページを読む際には，ユーザが知らない語（非既知語）が読解を妨げる原因の一つとなる．この問題に対処するためには，Web ページ中にある非既知語の語義を高速に表示する方法が挙げられる．

Web ページ中にある英単語を素早く辞書で引いて対応する辞書の項目（語義）を閲覧するためのインターフェースとして，Web ページを改変し，ページ中の非既知語をクリックしたりマウスオーバーしたりすることで，辞書の項目を表示するシステムが提案されてきた．このようなシステムを，本稿では，**語義注釈システム**と呼ぶ．

図 1 に挙げる **pop 辞書**[⋆1]は，日本語母語話者向け語義注釈システムの先鞭をとったシステムで，マウスオーバーすると語義をポップアップで表示する．また，**popIn** では，[⋆2] 選択した非既知語の語義を Web ページ中に埋め込んでいる．

語義注釈システムでは，ユーザがクリックした単語を記録することにより，ユーザの非既知語のログが蓄積される．このログを，本稿では**単語クリックログ**と呼ぶ．単語クリックログは，読解の障害となる非既知語のリストであるので，読解支援にとって有用な情報であると考えられる．既存の語義注釈システムでは，単語クリックログは活用されてこなかったが，単語クリックログを解析することにより，読解の障害となる非既知語を予測して予め語義を付与することで，ユーザはより高速に非既知語の語義を参照することが可能となる．

筆者らは，ユーザの回答パターンが記録されている単語クリックログから学習することによって，既存の語義注釈システムにユーザの語彙を予測する機能を付加したシステム **SocialDict**[⋆3]を提案している[3)]．本システムは，非既知語を自動的に予測し，その語に語義の

†1 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo
†2 東京大学情報基盤センター
Information Technology Center, The University of Tokyo
⋆1 http://www.popjisyo.com/
⋆2 http://www.popin.cc/
⋆3 http://www.socialdict.com/

注釈を付与する．ユーザが本システムにログインし，本システムを通して Web ページを閲覧した図が**図 2** である．赤く着色された部分が非既知と判別された部分であり，語義注釈が付与されている．黄色く着色された部分が既知と判別された部分である[⋆1]．

SocialDict は，任意の Web ページ全てについて適用可能であり，Gmail アカウントを用いてログインした後に，対象としたい Web ページの URL の前に `http://www.socialdict.com/` をつけてブラウザでアクセスすることで，語義注釈の処理を施すことができる．[⋆2]

しかし，SocialDict を 10ヶ月運用したところ，次のような問題点が浮かび上がってきた．

（1） 英文 Web ページを読むために，SocialDict にログインし URL を SocialDict に入力すること自体が，ユーザに取って手間となり，ユーザビリティを低下させている．

（2） Web ページの解析に時間がかけられないため，高度な解析を必要とするような素性（特徴量）を識別に利用することが難しい．

そこで本稿では，任意の英文 Web ページを対象とすることを犠牲にする代わりに，自然言語処理分野で重要な研究対象となっている Wikipedia に特化したシステム NEWikipedia を提案する[⋆3]．Wikipedia に特化させることにより，ユーザが要求する Web ページは Wikipedia に限定されるため，ユーザは特に URL の入力を意識せずに，Wikipedia と同様に NEWikipedia を使用することができるため，ユーザビリティを向上させることが可能となる．実際に，英語版 Wikipedia の項目の URL の *wikipedia* の前に *ne* を付け加えることで，そのまま利用出来るようにした[⋆4]．また，Wikipedia は事前にダウンロードすることが可能であるため，時間をかけて深い解析を行い，その結果を判別の素性に用いることが可能である．実際に，本稿では，Wikipedia に LDA を適用して得られた確率値を素性に用いることを試みる．

図 1 pop 辞書での注釈の例．

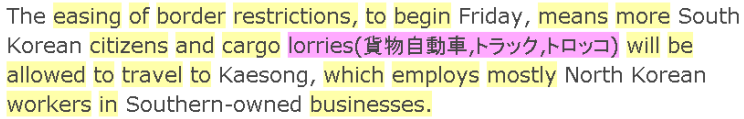

図 2 SocialDict での注釈の例

⋆1 本稿では説明のために既知と判断された語も着色しているが，筆者らの過去の発表の質疑応答において，この場合は着色しない方がよいという意見もあった．
⋆2 例：`http://www.socialdict.com/http://www.cnn.com/`
⋆3 http://en.newikipedia.org/

2. システムの構造

本節では，提案する語義注釈システムの構造について説明する．**図 3** に提案するシステムの構造を図示する．

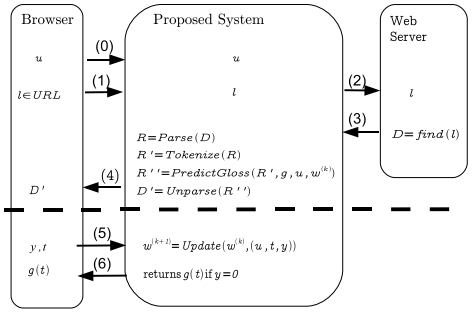

図 3 提案するシステムの構造

(0) ユーザはユーザ識別子 u を本システムに渡し，システムにログインする．SocialDict では，Gmail アカウントを用い，パスワード入力を伴なうログインを必要としており，ユーザビリティを低下させていた．NEWikipedia では，ブラウザをユーザとみなすことで，ログインなしでシステムを使用することを可能にした．

(1) ユーザは，ブラウザを通じて $l \in URL$ を SocialDict に渡す．URL は URL の集合とする．SocialDict では，直接 URL を指定する必要があり，これがユーザにとって手間となっていた．NEWikipedia では，対象を Wikipedia に限定されているため，前述のように少し URL を書き換えるだけで，URL を指定することができるようになった．

(2) システムは，渡された l が指し示す **Web サーバ** (Web Server) にアクセスする．

⋆4 例：`http://en.wikipedia.org/wiki/Main_page` に対して，`http://en.newikipedia.org/wiki/Main_page`

(3) Web サーバは l を受け取り，l に対応する Web ページ D を探索し（$D = find(l)$），システムに返す．

(4) システムは，D に注釈をつけて返す．この処理については本文を参照．

Web ページは，**図 4** (a) に示すように，テキストを葉，葉以外のノードをタグとするような木構造で表現することが可能である．全ての Web ページの集合を Dom_D，全ての木構造の集合を Dom_T と表す．Web ページ $D \in Dom_D$ を受け取り，木構造 $R' \in Dom_T$ を返す処理を $R' = Parse(D)$ と書く．逆に木構造 $R'' \in Dom_T$ を受け取り Web ページ $D' \in Dom_D$ を返す処理を $D' = Unparse(R'')$ と書く．

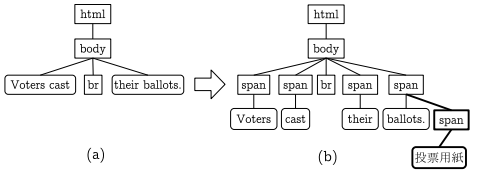

図 4 (a) Web ページの木構造, (b) ブラウザに返却される木構造．太線部分が付与される注釈の例．

図 3(4) では，図 4(a) に対して，**トークン化**と**予測機能による注釈**を行い，図 4(b) のように変換する．これらを，それぞれ **Tokenize** と **PredictGloss** という 2 つの，木構造を取り木構造を返す関数で表現する．**Tokenize** は木構造 $R \in Dom_T$ を受け取り，R の葉であるテキストをトークン化した木 $R' \in Dom_T$ を返す．図 4(a) をトークン化したものが，図 4(b) の赤字部分を除いた木構造である．図 4(b) 中でトークン化されたテキストの親タグとなる <**span**> では，クリック時に辞書を引き，語義を受け取る動作（図 3(5),(6) にそれぞれ相当）を実現するプログラムが JavaScript で記述され，埋め込まれる．

PredictGloss は木構造 $R' \in Dom_T$，注釈関数 g，ユーザ識別子 u，判別器の重み $w^{(k)}$ を受け取り，R' の葉に対して，ユーザ u の非既知語 t を $h(u, t, w^{(k)})$ の符号で判断し，t のみに注釈 $g(t)$ をつけて返す．ただし，R' はトークン化されていると仮定する．注釈関数 g は，トークン $t \in T$ を受け取り，t に注釈をつけた文字列 $g(t)$ をつけて返す関数である．

図 3(5), (6) では，**AJAX**（asynchronous JavaScript and XML）を用いてブラウザとシステムが通信を行う[⋆1]．

(5) D' 中のトークン t が最初にクリックされると，(4) での予測が訂正されたと判断し，単語の既知・非既知の情報 y をシステムに送出する．(4) で既知と判断されたトークン t がクリックされれば，非既知（$y = 0$）が送出される．(4) で非既知と判断されたトークン t がクリックされれば，既知（$y = 1$）が送出される．

(6) もし $y = 0$，すなわち，非既知の場合は，システムは t に対応する注釈 $g(t)$ を返し，$g(t)$ がブラウザで表示される．

(u, t, y) のデータの組は，判別器の重みベクトル $\boldsymbol{w}^{(k)}$ を更新するのに使用される．$Update(\boldsymbol{w}^{(k)}, (u, t, y))$ の詳細については，§3 で述べる．SocialDict，NEWikipedia，Web アプリケーションに特化したクラウド計算機環境である Google App Engine (GAE)[⋆2] 上で動作するように実装した．

3. 予測手法

項目反応理論[5)⋆3]（item response theory, IRT）は，人間の能力を測定するテストの設計に使用される確率的なモデルの総称である．TOEFL (Test of English as a Foreign Language) をはじめとする既存の言語テストの設計にも使用されているため，本研究でも項目反応理論を用いることが妥当であると考えられる．

項目反応理論は，テスト結果を入力として受け取る．テストは，$|T|$ 個の項目（設問）から構成されるとし，項目の集合を T と書く．被験者の集合を U とし，被験者数を $|U|$ と書く．被験者 $u \in U$ の項目 $t \in T$ に対する反応を $y \in Y$ とすると，(u, t, y) の組が 1 件のテスト結果となる．ただし，Y は，反応の種類の集合である．以上より，テスト結果は，その件数 N 件とすると $\{(u_n, t_n, y_n) | n \in \{1, \ldots, N\}\}$ と表すことが可能である．これが項目反応理論への入力となる．本研究では，単語クリックログをテスト結果 $\{(u_n, t_n, y_n) | n \in \{1, \ldots, N\}\}$ とみなす．ユーザ $u_n \in U$ の，文書中の個々の単語 $t_n \in T$ に対する反応を $y_n \in Y$ とみなす．Y は，本研究では，$Y = \{0, 1\}$ であるような二値変数とする．$y_n = 1$ のとき，ユーザ u_n は単語 t_n を知っている（既知）とし，$y_n = 0$ のとき，ユーザ u_n は単語 t_n を知らない（非既知）とする．

本研究では，項目反応理論のうち最も単純な Rasch モデルを，次のように改良して用いた．Rasch モデルでは，$P(y_n = 1 | u_n, t_n) = \sigma(\theta_{u_n} - d_{t_n}) p(\theta_{u_n}, d_{t_n})$ を最尤推定する．た

⋆1 この通信には，**jQuery** と呼ばれる JavaScript ライブラリを用いている．`http://semooh.jp/jquery/`

⋆2 `http://code.google.com/intl/ja/appengine/`

⋆3 日本語ではその他，**項目応答理論**，**テスト理論**などとも呼ばれる．

だし，$\sigma(x) = \frac{1}{1+exp(-x)}$ はロジスティックシグモイド関数であり，$p(\theta_{u_n}, d_{t_n})$ は事前分布である．また，θ_{u_n} は被験者 u_n の能力パラメータで θ_{u_n} が高いほど，被験者 u_n の正答率が増加する．また，d_{t_n} は項目 t_n の難易度パラメータで d_{t_n} が高いほど，被験者 u_n の項目 t_n に対する正答率が低下する．図 3 における $PredictGloss$ の内部で使用される判別関数 h は，$h(u_n, t_n, w^{(k)}) = \log P(y_n = 1|u_n, t_n; w^{(k)}) - \log P(y_n = 0|u_n, t_n; w^{(k)})$ と定義され，$h(u_n, t_n, w^{(k)}) \geq 0$ のとき既知，$h(u_n, t_n, w^{(k)}) < 0$ のとき非既知と判定される．ここで，予測精度を向上させるために，単語の難しさに関する素性を以下のように導入した．$\boldsymbol{e}_u$ を u 番目の要素のみ 1 で他は 0 のサイズ $|U|$ のユニットベクトル，$\boldsymbol{e}_t$ を t 番目の要素のみ 1 で他は 0 のサイズ $|T|$ のユニットベクトルとする．すると，尤度を，重みベクトル $\boldsymbol{w}_{rasch} = (\boldsymbol{\theta} \ \ \boldsymbol{d})^{\mathrm{T}}$ と特徴量ベクトル $\boldsymbol{\phi}_{rasch}(u,t) = (\boldsymbol{e}_u \ \ \boldsymbol{e}_t)^{\mathrm{T}}$ を用いて，数式 (1) と表すことができる．ただし，$\boldsymbol{\theta} = (\theta_1, \ldots, \theta_u, \ldots, \theta_{|U|})$，$\mathbf{d} = (-d_1, \ldots, -d_t, \ldots, -d_{|T|})$ である．

$$\begin{aligned} P(y_n = 1|u_n, t_n) &= \sigma\left(\theta_{u_n} - d_{t_n}\right) \\ &= \sigma(\boldsymbol{w}_{rasch}^{\mathrm{T}} \boldsymbol{\phi}_{rasch}(u_n, t_n)) p(\boldsymbol{w}_{rasch}) \end{aligned} \quad (1)$$

数式 (1) において，重みベクトル $\boldsymbol{w}_{rasch}$ を $\boldsymbol{w}_{LR} = (\boldsymbol{\theta} \ \ \boldsymbol{d} \ \ \boldsymbol{w}_a)^{\mathrm{T}}$ に，特徴量ベクトル $\boldsymbol{\phi}_{rasch}$ を $\boldsymbol{\phi}_{LR}(u,t) = (\boldsymbol{e}_u \ \ \boldsymbol{e}_t \ \ \boldsymbol{\phi}_a)^{\mathrm{T}}$ に置き換えることにより，$\boldsymbol{\phi}_a$ に素性を追加することが可能となる．また，事前分布 $p(\boldsymbol{w}_{rasch})$ には，$\boldsymbol{w}_{rasch}$ を m 次元のベクトルとすると，m 次元の多次元ガウス分布 $\mathcal{N}(\mathbf{0}, \frac{1}{C}\boldsymbol{I})$ がよく用いられ，本稿でも後述の実験でこれを用いている．

次に，パラメータの推定手法について説明する．パラメータを一回更新する際に，データセット全体（この場合は，$\{(u_n, t_n, y_n)|n \in \{1, \ldots, N\}\}$）に対して最適化を行うパラメータ推定手法をバッチ学習法という．Rasch モデルのパラメータを MAP 推定や最尤推定を用いて推定する方法は，バッチ学習法に分類される．§2 において，$Update$ 関数にバッチ学習法を用いると，ユーザがクリックするたびにデータセット全体を参照し最適化を行う必要が生じる．つまり，蓄積された単語クリックログのデータ数 N に対して，一回のクリックごとのパラメータ更新に $O(N)$ 以上の計算が必要となる．従って，単語クリックログのデータ数 N が増加すると，いずれ実用的な時間で学習が行えなくなるため，実運用環境ではバッチ学習は適さないことがわかる．この問題を解決するために，精度を犠牲にする代わりにパラメータを逐次的に推定し，一回のクリックごとのパラメータ更新が $O(1)$ で終了する逐次学習法が提案されている．Rasch モデルに対しては，逐次学習法の 1 つとして Stochastic Gradient Descent (SGD)[1] が利用できる．SGD では，最尤推定における n に関する和を省略して，$Update$ 関数を次のように定める．

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \eta_k \nabla E_n(\mathbf{w}^{(k)}) \quad (2)$$

数式 (2) は，$\eta_k = \frac{1}{\lambda(k+k_0)}$ であるときに収束する．ただし，λ, k_0 は非負の定数である．SGD は，逐次学習法であり，逐次的にデータが蓄積される本システムに適すると考えられるため，実運用している SocialDict，NEWikipedia では，共にこれを用いている．

4. 実験と考察

実験では，テストセットに対する既知/非既知の予測精度を測定した．単語クリックログが延べ語数で N_0 個蓄積されたところに，ユーザが 1 人新規にシステムを使い始め，この新規ユーザが異なり語で N_1 個の単語の既知/非既知を入力したと想定し，そのユーザのテストセットに含まれる単語のうち異なり語で何% について既知/非既知を当てられたかを 1 人の精度とした．本実験では，入力データの順序を問題とせず，素性を追加することによって予測精度がどのように変化するかを測定したいので，バッチ学習を用いて実験を行った．

精度評価のために，1 人，後述する **SVL12000** という語彙集のうち 11,999 語について，5 段階[⋆1]の自己申告形式で回答させる方法で，被験者（東京大学修士大学院生 16 人）の語彙力を測定した．このうち，意味を確実に知っている場合のみを既知の場合とし，残りを非既知の場合とした．

11,999 語を，1400 語のデベロップメントセット，9999 語のテストセットと，残りの 600 語に重なりがないように分割した．予め蓄積されていると想定している N_0 個の単語クリックログに，この 600 語のうち $N_1 \in \{10, 30, 100, 300, 600\}$ 語を新規のユーザがクリックしたと想定し，計 $N_0 + N_1$ 個のデータを訓練データとした．この予め蓄積された単語クリックログとしては，実際にシステムを運用して得られた単語クリックログから，正例と負例の数を 300 個ずつに調整した $N_0 = 600$ 個のデータを用いた．§3 で述べた事前分布の C パラメータは，$\{0.0315, 0.125, 0.5, 1.0, 2.0, 8.0, 32.0, 128.0, 512.0, 2048.0, 32768.0\}$ から，デベロップメントセットに対して最適な値を選択した．

追加する素性を変化させることにより，3 種類の実験設定を用意し，それぞれ **IRT**，**Basic** と **Basic+Wiki** と名付けた．

IRT　§3 で述べた IRT をそのまま使用している．

⋆1 単語を知らない度合いの大きい順に，見たこともない，見たことがある気がする，確実に見たことはあるが意味は知らない／覚えたことがあるが意味を忘れている，意味を知っている気がする／意味が推測できる，意味を確実に知っている．

Basic　§3 で述べた方法により **IRT** に加えて，トピックを無視した一般的な単語の難易度を表す素性として，**Google 1-gram** と **SVL12000** という 2 種類の素性を追加している．**Google 1-gram** は，約 1 兆ページの Web ページ中の英単語の頻度であり，人手を介していない[2]．**SVL12000** は，基本的な語彙 12,000 語に対し，英語母語話者を含むチームが人手で 12 段階の難易度をつけた語彙リストである[4]．

Basic+Wiki　**Basic** に加えて，トピックごとの単語の難易度を表す素性を追加している．英語版 Wikipedia 約 132 万項目に対して，トピック数 300 の LDA (Latent Dirichlet Allocation) を実行して，各単語 t についてトピック z ごとの単語の出現確率の推定値 $p(t|z)$ の確率値を得た．この各単語についてトピック数個ある確率値を，トピックごとの難易度を表す単語の素性とみなし，§3 に述べた方法で素性に追加している．

確率値 p は，全て $-\log p$ の形で素性に追加した．頻度も，全頻度で正規化して確率値に変換し，この方法を用いて素性に追加した．

表 1　各素性の予測精度 (%)．

	$N_1 = 10$	30	100	300	600
IRT	**66.70**	**66.64**	67.36	67.64	67.64
Basic	64.18	66.09	**71.77**	**76.05**	**78.62**
Basic+Wiki	48.10	54.33	58.84	67.09	71.23

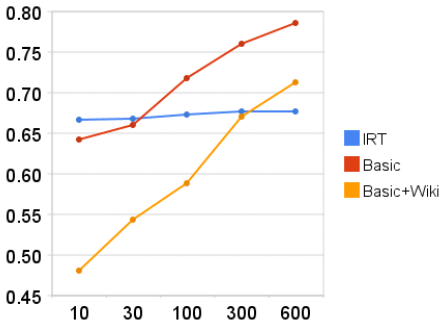

図 5　実験結果．横軸は N_1．縦軸は本文で定義した精度．

表 4，**図 5** に実験の結果を示す．まず，素性を追加した **Basic** は N_1 が増加するのに対して，ベースラインとなる **IRT** は，N_1 に対してほぼ平坦となっており，ほとんど学習の効果が現れていないことがわかる．この理由は，次のように説明できる．まず，単語の難易度を表す素性を追加した **Basic** では，この追加素性に対する重みが学習され，テストセットの単語が訓練データセットに現れていない場合であっても，その単語の **Basic** に属する素性の値を通じて単語の難易度を概算し，予測することが可能である．しかし，単語の難易度を表した素性を全く追加していない **IRT** では，テストセットの単語が訓練データセットに現れていない場合，判別器はその単語の難易度を知る術がないので，予測することは不可能となる．今回の実験設定では，特に，テストセットの単語数に対して訓練データセットのサイズが小さいので，この現象が顕著に現れる．

次に，さらに素性を追加した **Basic+Wiki** は，**Basic** より精度が劣っているものの，N_1 に対する精度の向上率は大きい．この理由は，**Basic+Wiki** では，**Basic** に加えて，各トピックごとに 300 個もの単語の難易度を表す素性があるため，訓練データ数が少ないため重みの学習が追いついていないと考えられる．従って，N_1 を増加させると，**Basic** より精度が向上すると推測される．しかし，本研究の目的では，N_1 を増加させるということは，1 人のユーザがより多くの単語をクリックしなければならないということであるため，望ましいことではない．**Wiki** の素性を追加した場合でも，小さい N_1 で十分な精度を達成することが，今後の課題である．

5. 結論と今後の課題

筆者らは，任意の英文 Web ページに対して，クリックログの解析を基にユーザ知らない英単語をユーザの英語力に応じて予測することで，Web ページの読み込み時にそれらの語に対して訳を予め付与する，ユーザ参加型の読解支援システム SocialDict を提案している．しかし，SocialDict では，ログインや URL の入力がユーザに取って手間となりユーザビリティが低下する問題，対象とする Web ページに対する深い解析を必要とする特徴量を予測に利用することが難しいという問題があった．任意の英文 Web ページを対象とすることを犠牲にする代わりに，Wikipedia に特化したシステム NEWikipedia を提案する．これにより，ログインや URL の入力を不要としてユーザビリティを向上させ，LDA を用いて得られたトピックごとの単語頻度の特徴量を判別に利用することが可能となる．

予測には，TOEFL などの言語テストに使用されている項目反応理論を使用することが妥当であると考え，その一種である Rasch モデル (**IRT**) を使用した．さらに，Rasch モデルを拡張し，**Basic** 素性を追加して予測精度が向上することを実験によって示した．一方，実

Vol.2010-HCI-138 No.4
2010/5/14

運用環境では，単語クリックログが逐次的に与えられるため，Rasch モデルに対してバッチ学習法ではなく，逐次学習法である SGD を用いている．

しかし，同じく実験によって，さらにトピックごとの単語の難易度の素性を追加した **Basic+Wiki** 素性では，実験設定である訓練語数 $N_1 \leq 600$ 語の範囲では，精度は **Basic** より向上しなかった．この理由は，訓練語数が少なすぎ，重みの更新が追いついていないためであると考えられる．しかし，本研究の目的では，N_1 を増加させるということは，1 人のユーザがより多くの単語をクリックしなければならないということであるため，望ましいことではない．

今後の課題としては，**Wiki** の素性を追加した場合でも，小さい N_1 で十分な精度を達成することが挙げられる．

謝辞 評価実験の実験協力者の皆様，大規模な Wikipedia コーパスに対して LDA を適用してくださった吉田和弘氏に，謹んで感謝の意を表する．

参 考 文 献

1) Bishop, C.M.: *Pattern recognition and machine learning*, Springer (2006).
2) Brants, T. and Franz, A.: *Web 1T 5-gram Version 1*, Linguistic Data Consortium, Philadelphia (2006).
3) Ehara, Y., Shimizu, N., Ninomiya, T. and Nakagawa, H.: Personalized reading support for second-language web documents by collective intelligence, *2010 International Conference on Intelligent User Interfaces (IUI 2010)*, pp.51–60 (2010).
4) SPACE ALC Inc.: Standard Vocabulary List 12,000 (1998). Data available at `http://www.alc.co.jp/goi/PW_top_all.htm`.
5) 豊田秀樹：項目反応理論 [理論編]-テストの数理-，朝倉書店 (2005).