2016年度 卒業論文

ソーシャルリーディングによる
学習補助システムの開発

大阪産業大学 デザイン工学部 情報システム学科
情報教育システム研究室

13H063 中谷元

目次

1　はじめに

プログラミングなどの学習の際に、いわゆる技術書を読む機会がある。しかし専門的な内容の為、書籍の内容を網羅することは難しい。読者の周囲に知識または経験が豊富な人物が存在する場合には、指導を仰ぐなどが出来る。だがそういった人物が存在しない場合、誤った解釈をすることや内容を理解出来ないことがある。

Web 上に技術書の感想が公開されている場合もあるが、情報がひとところに集まっているわけではない。また情報を公開した人物が誤った解釈をしている場合もある。このような問題を解決するためには、技術書についての情報をひとところに集め、読者同士で技術書について質疑応答や議論を行なえる場が必要であると考えた。

第 2 章ではこの問題をふまえて本研究の目的について述べる。第 3 章では類似するサービスについて述べる。第 4 章では本研究で開発したシステムの概要を述べる。第 5 章では本研究で開発したシステムの結果をその考察とともに述べる。第 6 章には研究の成果とともに今後の課題についてまとめる。

2 目的

Web で技術書について検索した場合、書評などが公開されている場合がある。しかしブログなどで公開されていることが多く、情報がひとところに集まっていない。筆者に連絡が付く保証は無く、筆者が誤った解釈をし情報を公開していることもある。

そこで本研究ではソーシャルリーディング [1] を活用して、技術書を用いた学習効率の向上を補助するシステムを提案する。読者の周囲に知識または経験が豊富な人物が存在しない場合でも、インターネットを介して内容の理解に繋がるシステムを開発する。

3　類似のサービス

本章では本研究で開発するサービスの先行・類似のサービスについて述べる。

3.1　ポピュラー・ハイライト

Kindle が提供するサービス。Kindle ストアで購入した書籍の場合、他のユーザーによってハイライトされた文章が表示される。ポピュラー・ハイライトが有効な状態のスクリーンショットを図 1 に示す。

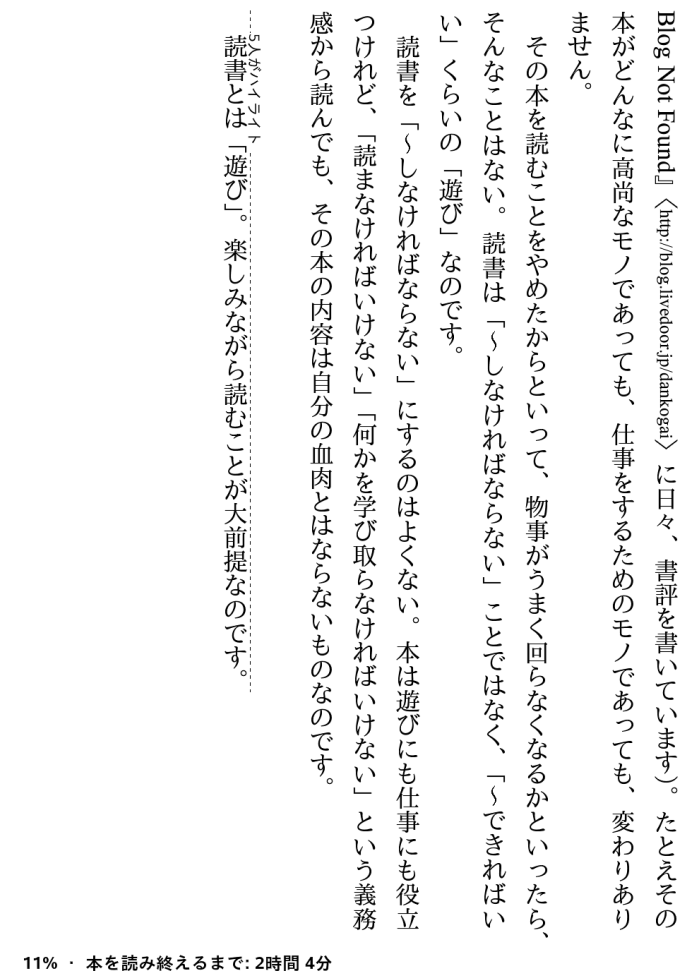

図 1　『空気を読むな、本を読め。』[2] を Kindle Paperwhite で読書中のスクリーンショット。他のユーザーがハイライトした箇所が表示される。

3.2　ブクログ

Web 上に個人の本棚を作るサービス。本を「読みたい」「いま読んでる」「積読」「読み終わった」等で分類して管理出来る。フォロー機能があり、他のユーザーの本棚を確認出来る。ユーザーの本棚の様子を図 2 に示す。

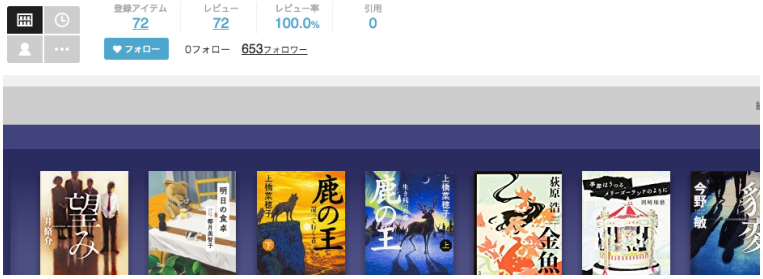

図 2　本棚の様子。条件を指定することで絞り込みが可能。

3.3　感想ライブラリー

読書感想文を投稿するサービス [3]。各書籍に対して読書感想文を投稿することが出来る。書籍単位でコメントをすることは出来るが、感想分単位でのコメントは出来ない。投稿された読書感想文の様子を図 3 に示す。

図 3　書籍に対し複数の読書感想文が掲載される。

3.4　Stack Overflow

プログラミング技術に関するナレッジコミュニティ [4]。質問も回答も匿名で投稿することが出来て、ユーザーには信用度というプロパティがある。信用度は質問や回答を投稿したり、他のユーザーに発言を評価されることで変動する。質問に寄せられる回答の様子を図 4 に示す。

図 4　質問に対し他のユーザーから回答を得ることが出来る。

4　ソーシャルリーディングによる学習補助システム

本システムではユーザーが登録された書籍に対し、技術書についての内容を投稿する。本研究では投稿内容のことをブックレポートと呼ぶ。

類似するサービスで挙げた感想ライブラリー [3] では、書籍毎にコメントを残すことは出来たが、これでは感想文単位での質疑応答や議論が出来ない。このことから本システムではブックレポート毎にコメントを残せるようにする。これにより、ブックレポート毎に質疑応答や議論を行えるようにする。また Stack Overflow [4] では、質問に対する回答を投稿出来るが、特定の回答に対する返信を行えない。このことからコメント機能に加えリプライ機能を用意する。これにより、より議論を行ないやすくする。

これらのことから本システムでは、類似するサービスと比べより学習に長けたシステムである。読者の周囲に知識または経験が豊富な人物が存在しない場合や、周囲の人物に指導を仰ぐことが出来ない場合でも、読者の学習を補助出来る。

4.1　ソーシャルリーディング

ソーシャルリーディングとは、SNS[*1]などのメディアを利用して、読者が書籍の中で感銘を受けた部分を紹介したり、読了後の感想を他のユーザーと共有し語り合うといった、読書にまつわる情報を共有し、読書体験の向上や読書体験を通じた人と人との繋がりを実現すること [5]。いわば、書評を通じたコミュニケーションのこと。

4.2　システムの動作

本システムではユーザー自身にブックレポートを投稿してもらう。投稿対象の技術書が無い場合は、書籍の登録を行なう。ブックレポートを投稿した後、他のユーザーからのコメントを待ち、質疑応答や議論を介して、内容の理解に繋げる。また、自分以外のブックレポートに対しコメントを残すことで、他のユーザーの理解を補助する。

本システムはユーザー認証をしない場合でも、ブックレポートの閲覧が可能とする。未認証ユーザーには書籍の登録やブックレポートの投稿、コメントの投稿などに制限がある。これにより未認証でもブックレポートの閲覧のみを可能とする。

またユーザー毎に信用度という値を設定する。これは発言に重みを付ける目的の為に設定した値である。これによりブックレポートに寄せられたコメントの情報を、ユーザーが判断する際の補助となる。信用度は特定の行動[*2]を行なうことにより変動する。本システムのフローチャートを図 5 に示す。

[*1] ソーシャルネットワーキングサービス

[*2] 4.3.4 を参照

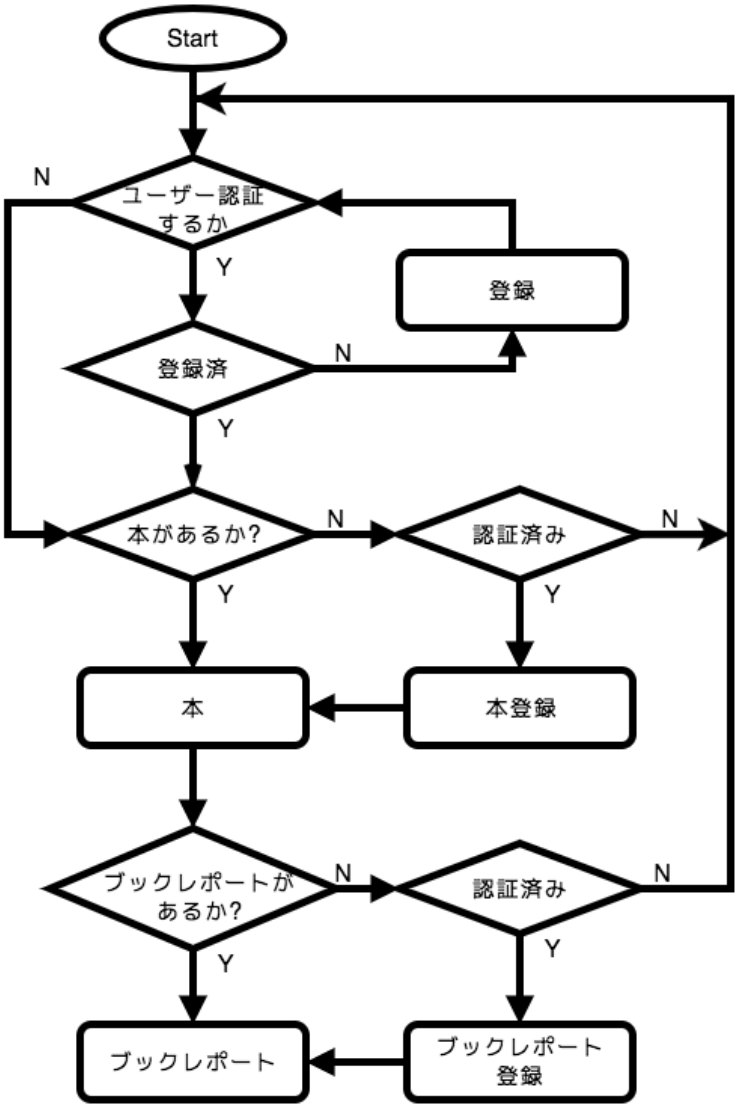

図 5　本システムのフローチャート

未認証の場合でも、Web ページ上部のナビゲーションバーから、書籍一覧やログイン、ユーザー登録画面に遷移出来る。図 6 に Web ページ上部のスクリーンショットを示す。

図 6　Web ページ上部のナビゲーションバー。ユーザー認証していない場合でも書籍一覧からブックレポートを閲覧出来る。

書籍一覧では、登録済の書籍一覧が表示される。投稿したいブックレポートの対象書籍が存在しない場合は、書籍登録を行う。書籍登録の場合は、まず ISBN[*3]を元に書籍を検索する。検索結果が正しかった場合は、書籍登録が完了する。図 7 に書籍一覧画面、図 8 に ISBN 検索画面、図 9 に書籍登録画面を示す。

[*3] International Standard Book Number

図 7　登録済の書籍一覧の画面

図 8　書籍登録前に ISBN で検索をする。

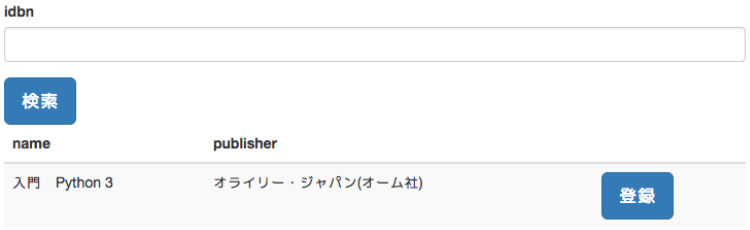

図 9　検索結果が正しい場合登録

各書籍にはリンクが用意されている。リンク先に遷移すると、書籍に対応するブックレポートの一覧が表示される。他のユーザーのブックレポートを閲覧する場合は、各ブックレポートのリンクボタンを押す。ブックレポートを投稿する場合は、ブックレポートの登録を行う。図 10 にブックレポート一覧画面、図 11 にブックレポート登録画面を示す。

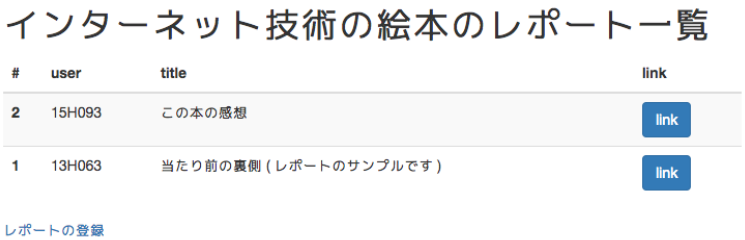

図 10　登録済のブックレポート一覧の画面

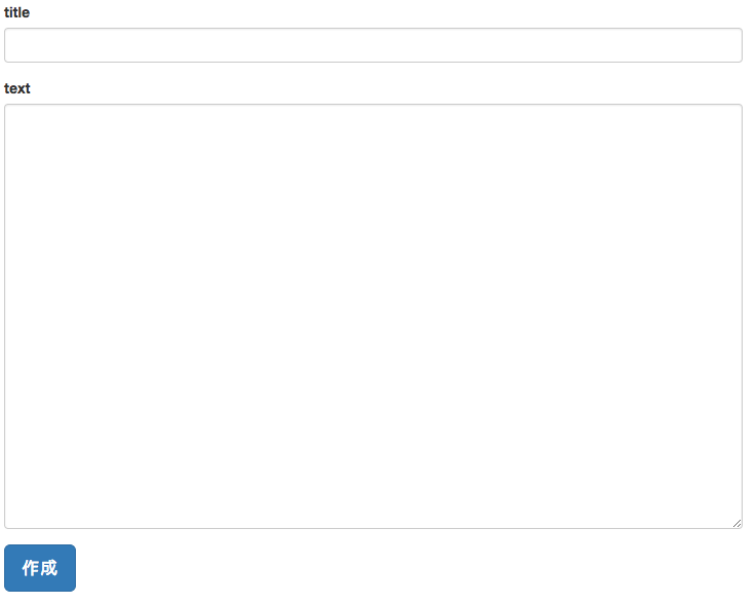

図 11　ブックレポート登録画面

登録済のブックレポートでは、ブックレポートを投稿したユーザー、またはコメントを残したユーザーの評価を行える。評価されたユーザーは信用度が変動する。

リプライ機能によって、特定のコメントに返信を行なう事が出来る。これは、より質疑応答や議論を行いやすくする為の機能である。リプライコメントは通常のコメントと違い、リプライ先へのコメントがリンクとして挿入される。図 12 にブックレポート画面、図 13 に画面のコメント部分、図 14 にリプライ部分の画像を示す。

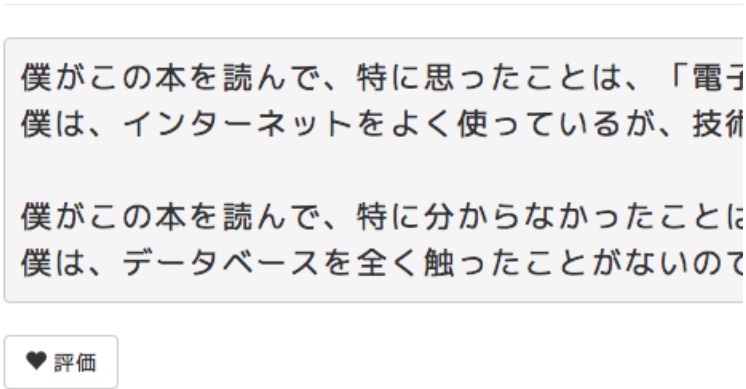

図 12　ブックレポート画面

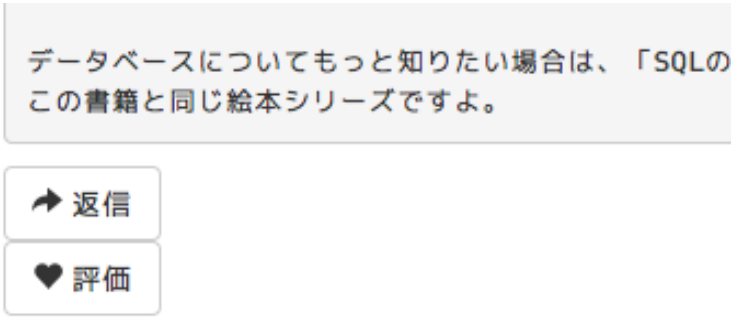

図 13　ブックレポート画面のコメント部

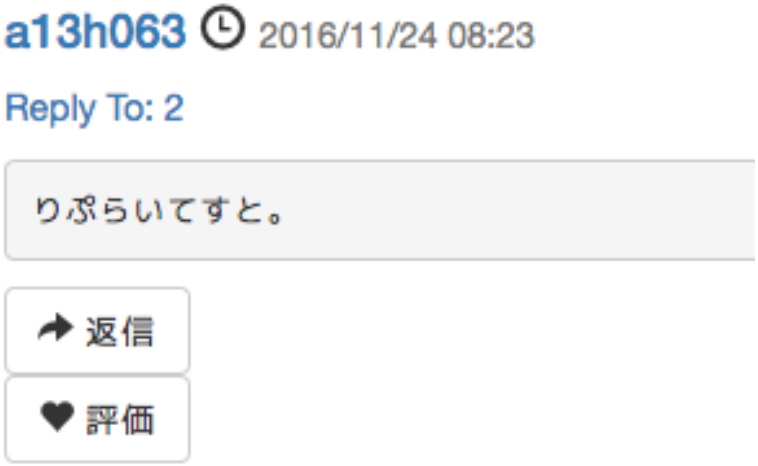

図 14　ブックレポート画面のリプライ部

本システムではユーザー名がすべてリンクになっており、ユーザー固有のページ、いわゆるマイページに遷移する。マイページではユーザー名、信用度が確認出来る。図 15 にマイページを示す。

図 15　マイページ

4.3　仕様

4.3.1　構成

本システムは、Python 製 Web アプリケーションフレームワークの Django を使い構築した。Django には Web アプリケーションで良く利用され、本システムでは必須となる以下の機能が組み込みで用意されている。

- ユーザー認証
- セッション
- セキュリティ

本システムではそれらの機能を拡張するように開発をした。データベースにはリレーショナルデータベースを利用する。OSS[*4]の PostgreSQL を使用した。

Web サーバには Apache を使用し、モジュールの mod_wsgi を使用することで、wsgi サーバを立てる必要をなくした。

- Apache
  - mod_wsgi
- Python
  - Django
- PostgreSQL

4.3.2 動作

本システムの動作として、表 1 に URL の構成を示す。Django は標準では PUT と DELETE の 2 つの HTTP method に対応していない為、GET と POST のみを利用した。

表 1　本システムにおける PATH の一覧

HTTP Method	PATH	概要
GET	/	トップページ
GET	/users/signup/	ユーザー登録ページ
POST	/users/signup/	ユーザー登録
GET	/users/signin/	サインインページ
POST	/users/signin/	サインイン
GET	/users/mypage/<user_id>/	マイページ
GET	/books/	書籍一覧
GET	/books/new/	書籍登録ページ
POST	/books/new/	書籍登録
GET	/books/<book_id>/	書籍概要
GET	/books/<book_id>/reports/	ブックレポート一覧
GET	/books/<book_id>/reports/new	ブックレポート登録ページ
POST	/books/<book_id>/reports/new	ブックレポート登録
GET	/books/<book_id>/reports/<report_id>	レポート概要
POST	/books/<book_id>/reports/<report_id>	コメント登録

また Trailing Slash が付与されていない URL に対しては、Trailing Slash 有りの URL にリダイレクトさせるようにしている。こうすることで、URL の末に Slash があるか否かで挙動が変わることを回避した。

名詞の複数形では、データ一覧を参照することが出来る。ソート機能等は HTTP パラメータにソート時に利用する key を渡すことで、同 URL で挙動を変えることが出来る。

名詞の単数形では、各データの primary key を続けることで、特定のデータにアクセス出来る。URL の検索には正規表現を用いることで、Primary key の桁数が増えても問題がない。

[*4] オープンソースソフトウェア

4.3.3 書籍登録

本システムでは書籍検索時に、「データベース日本書籍総目録」[6] に対して ISBN 検索を行う。こちらは ISBN10 桁[*5]でも検索が可能である。

ISBN での検索後に、ISBN を Primary key としてデータを保存することで、書籍の重複登録を回避する。「データベース日本書籍総目録」に対し検索をすると、著者名や価格を始めとした、多様な情報を取得出来る。本システムではその中から ISBN と書籍名、出版社を保存する。図 16 に ISBN 検索を行った結果画面、図 17 に検索結果後の書籍の詳細画面を示す。

図 16　書籍検索結果画面

図 17　検索結果の詳細画面

[*5] 2006 年 12 月以前は ISBN は 10 桁であった。2017 年以降から ISBN13 桁に変更された。

4.3.4 **信用度**

本システムでは投稿内容に重みを付ける為、信用度を設定した。投稿に重みを付けることにより、ブックレポート投稿者は、自身の投稿に寄せられたコメントに対し、コメント投稿者の信用度を加味して情報を得られる。

章 2 に、信用度の変動条件を示す。

表 2　信用度の変動条件

行動	評価対象	変動値
ブックレポートを投稿する	ブックレポート投稿者	+2
ブックレポートを評価される	ブックレポート投稿者	+5
コメントを評価される	コメント投稿者	+10

これらの評価値は類似サービスの 1 つである Stack OverFlow を参考にした。Stask OverFlow ではユーザーに信用度というパラメータを採用している。表 3 に Stack OverFlow における、信用度の変動条件を示す。

表 3　Stack OverFlow における信用度の変動条件

行動	評価対象	変動値
質問がプラス投票された	質問投稿者	+5
回答がプラス投票された	回答投稿者	+10

Stack OverFlow では上記以外の行動でも信用度が変動するが、今回参考にした項目は以上である。また Stack Overflow では信用度の下方評価もあるが、本システムでは上方評価のみである。

4.3.5 **リプライ**

本システムではブックレポート単位での、ユーザー同士による質疑応答や議論を円滑に行えるように、特定のコメントに対するリプライ機能を実装した。

リプライコメントは通常のコメントと違い、コメント内容の前にリプライ先のコメントへのリンクが挿入される。

5　結果と考察

本研究では技術書を用いた学習の際に、ソーシャルリーディングを活用して、技術書を用いた学習効率の向上を補助するサービスを提案した。そこでユーザーが自由に書籍、ブックレポートそしてコメントを投稿出来るシステムを開発した。以下に本研究で開発したシステムの考察を示す。

5.1　利点

5.1.1　情報の集約

Web 上に技術書についての書評などが公開されているが、ブログや特定 Web サービスを利用したものが多い。図 18 に Google で技術書の書評を検索した際のスクリーンショットを示す。

図 18　『オブジェクト指向でなぜつくるのか』[7] の書評を Google で検索した上位 3 件

この点について本システムでは技術書についての情報を集約した。書籍登録もユーザーが行えることで、利用者が増えるにつれ情報が増えることが期待出来る。

5.1.2　信用度による情報の重み付け

4.3.4 で示した通り、すべてのユーザーは同じ評価条件で評価される。ブログなどで公開されている書評では、書評の内容から情報の信用度を計り難い。本研究で提案するサービスでは、

- コメント自体の内容
- コメントを投稿したユーザーの信用度

の 2 点から情報の信用度を計れる。

投稿者がブックレポートに寄せられたコメントを、正しいか否か判断出来ない場合に、該当コメントを投稿したユーザーの信用度から、情報の真偽を判断する助けとなる。

匿名性の高い場での発言において、故意に投稿された誤った情報を、機械的に偽であると判断し難い。その為、信用度による発言の重み付けにより、類似サービスと比べより学習向けである。

5.2 問題点

5.2.1 下方評価

本システムでは信用度の下方評価がない。現状では誤った情報や、議論の妨害など他のユーザーに対し、マイナスな言動を行なった場合は、評価をしないことが前提にある。そういった際に、信用度を下方評価する機能が必要である。

5.2.2 匿名投稿

本システムでは未認証の場合、書籍登録、ブックレポート投稿、コメント投稿などの機能は出来ない。未認証の状態でブックレポートを閲覧し、発言をするには必ず認証しないといけない為、匿名での投稿が必要である。

5.2.3 ソーシャル性

本システムでは類似サービスのようなソーシャル性が高くない。

- フォロー機能
- ユーザー間の個別メッセージ

上記の様な SNS で使い慣れた機能があれば、より一層使いやすいサービスになる。例えば、フォロー機能があれば信用度の高い人をフォローしておくことで、その人の発言に注目することが出来るなど。

また他のサービスとの連携機能も不足している。現在の SNS ではサービス間の連携が豊かで、別の SNS などに情報を共有することが出来る。そういった機能があれば、サービス自体の認知度の向上に繋がる。

6　結論

本研究では、ソーシャルリーディングを活用することにより、技術書を用いた学習効率の向上を補助するサービスの提案を行ない、それらの機能を踏まえたシステムを開発した。

その結果、読者の周りに知識または経験の豊富な人物がいない場合などでも、インターネットを介して技術書についての知見を広めることが出来る。

信用度により発言に重みを付けた。また質疑応答や議論のしやすさを考慮してリプライ機能を付けた。類似サービスと比べより学習向けであるといえる。ブックレポートは自由形式での投稿であるため、技術書についての内容の要約や、疑問点を投稿するなどの自由がある。

本システムを用いることで、技術書を用いた学習の補助が出来るため、独学で学習する者などに貢献することが期待できる。

6.1　今後の課題

6.1.1　匿名投稿

匿名投稿機能を追加することで、試しにサービスを利用したいユーザーなどの、参入障壁を取り除くことが可能である。匿名投稿を可能にする場合、既存の投稿機能周りの改修も必要となる。

6.1.2　SNS 機能拡張・評価値見直し

他のユーザーの動向を追うことが出来るフォロー機能、またはそれに準ずる機能の実装。機能拡張に伴ない信用度の変動条件の見直しを改めることが必要になる。

6.1.3　NG 機能

投稿する内容に不適切な内容を含んでいた際に投稿を制限したり、不適切な行為を行なったユーザーに制限を課したり等の NG 機能が必要になる。

7 謝辞

本研究を進めていく上で、大垣 斉准教授に御指導及び御協力を戴きました。また本研究の検証に協力していただいた方々及び情報教育システム研究室のメンバ、研究室の OB の方々、team.andrew ML のメンバの方々には御助言を賜りました。

ここに深く感謝の意を表します。

参考文献

[1] ソーシャルリーディングとは - it 用語辞典 weblio 辞書. http://www.weblio.jp/content/socialreading, 2016.

[2] 小飼弾. 空気を読むな、本を読め。イースト・プレス, 2009.

[3] 感想ライブラリー. https://www.kanso-library.com/.

[4] Stack overflow. http://stackoverflow.com/.

[5]「ソーシャルリーディング」が開く読書の新世界. http://www.nikkei.com/article/DGXNASFK1901H_Z11C12A2000000/, 2012.

[6] データベース日本書籍総目録. http://www.jbpa.or.jp/database/.

[7] 平澤章. オブジェクト指向でなぜつくるのか 第 2 版. 日経 BP 社, 2011.

付録 A　ソースコード

A.1　./sotsuken/settings.py

Listing 1　設定

```
"""
Django settings for sotsuken project.

Generated by 'django-admin startproject' using Django 1.9.6.

For more information on this file, see
https://docs.djangoproject.com/en/1.9/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/1.9/ref/settings/
"""

import os
import socket

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))



# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/1.9/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'a-n7fo^or&g=em@pppn2%8+1d6dv&c==-$nh4@@fhcn^$ixyl6'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

STATIC_URL = '/static/'

HOSTNAME = socket.gethostname()

if 'debian' == HOSTNAME:
```

```
    DEBUG = False
    ALLOWED_HOSTS = ['*']
    STATIC_URL = '/ks132/static/'

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    # Third party's Apps
    'django_extensions',

    # Local Apps
    'users.apps.UsersConfig',
    'bookreports.apps.BookreportsConfig',
]

MIDDLEWARE_CLASSES = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'sotsuken.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
```

```
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
    {
        'BACKEND': 'django.template.backends.jinja2.Jinja2',
        'DIRS': [os.path.join(BASE_DIR, 'jinja2')],
        'APP_DIRS': True,
        'OPTIONS': {
            'environment': 'sotsuken.jinja2.environment',
        },
    },
]

WSGI_APPLICATION = 'sotsuken.wsgi.application'


# Database
# https://docs.djangoproject.com/en/1.9/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'sotsuken',
        'USER': 'sotsuken',
        'PASSWORD': 'sotsuken',
        'HOST': 'localhost',
        'PORT': 5432,
    }
}


# Password validation
# https://docs.djangoproject.com/en/1.9/ref/settings/#auth-password-validators
```

```
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.
            UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.
            MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.
            CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.
            NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/1.9/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/1.9/howto/static-files/

STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')

STATICFILES_DIRS = [
```

```
    os.path.join(BASE_DIR, 'bower_components'),

    os.path.join(BASE_DIR, 'static'),
]

STATICFILES_FINDERS = [
    'django.contrib.staticfiles.finders.FileSystemFinder',
    'django.contrib.staticfiles.finders.AppDirectoriesFinder',
]

AUTH_USER_MODEL = 'users.CustomUser'
```

A.2 ./sotsuken/urls.py

Listing 2 Urls

```
"""sotsuken URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/1.9/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  url(r'^$', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  url(r'^$', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.conf.urls import url,
        include
    2. Add a URL to urlpatterns:  url(r'^blog/', include('blog.urls'))
"""
from django.conf.urls import url, include
from django.contrib import admin
from django.views.generic import TemplateView

urlpatterns = [
    url(r'^admin/', admin.site.urls),

    # generic
    url(r'^$', TemplateView.as_view(template_name='home.jinja2'), name='home')
        ,
```

```
    url(r'^about/$', TemplateView.as_view(template_name='about.jinja2'), name=
        'about'),

    url(r'^users/', include('users.urls')),
    url(r'^books/', include('bookreports.urls')),
]
```

A.3 ./users/urls.py

Listing 3 users アプリケーションの Urls

```
# -*- coding: utf-8 -*-

from django.conf.urls import url

from . import views

app_name = 'users'

urlpatterns = [
    url(r'^signup/$', views.signup, name='signup'),
    url(r'^signin/$', views.signin, name='signin'),
    url(r'^logout/$', views.logout_view, name='logout_view'),
    url(r'^mypage/(?P<pk>[0-9]+)/$', views.MypageView.as_view(), name='mypage'
        )
]
```

A.4 ./bookreports/urls.py

Listing 4 bookreports アプリケーションの Urls

```
# -*- coding: utf-8 -*-

from django.conf.urls import url

from . import views

app_name = 'bookreports'

urlpatterns = [
    url(r'^$', views.BooksView.as_view(), name='books'),
    url(r'^new/$', views.book_new, name='book_new'),
    url(r'^(?P<pk>[0-9]+)/$', views.BookView.as_view(), name='book'),
    url(r'^(?P<pk>[0-9]+)/reports/$', views.reports, name='reports'),
```

```
    url(r'^(?P<pk>[0-9]+)/reports/new$', views.report_new, name='report_new'),
    url(r'^(?P<book_pk>[0-9]+)/reports/(?P<report_pk>[0-9]+)/$', views.report,
        name='report'),
    url(r'^(?P<book_pk>[0-9]+)/reports/(?P<report_pk>[0-9]+)/reputation/$',
        views.reputation, name='reputation'),
]
```

A.5 ./users/models.py

Listing 5 users アプリケーションのモデル

```
# -*- coding: utf-8 -*-

from django.db import models
from django.contrib.auth.models import AbstractUser


class CustomUser(AbstractUser):
    reputation = models.IntegerField(default=0)
```

A.6 ./bookreports/models.py

Listing 6 bookreports アプリケーションのモデル

```
from django.db import models
from django.utils import timezone

from users.models import CustomUser

DIGIT = 13


class Book(models.Model):
    id = models.BigIntegerField(primary_key=True)
    name = models.CharField(max_length=200)
    publisher = models.CharField(max_length=32)

    def __str__(self):
        return self.name

    @classmethod
    def valid_isbn(cls, isbn):
        return isbn.isdigit() and len(isbn) == DIGIT
```

```
class Report(models.Model):
    user = models.ForeignKey(CustomUser)
    book = models.ForeignKey(Book)
    title = models.CharField(max_length=128)
    text = models.TextField()

    def __str__(self):
        return 'user: {}, book: {}, title: {}'.format(self.user.username, self
            .book.name, self.title)


class Comment(models.Model):
    report = models.ForeignKey(Report)
    user = models.ForeignKey(CustomUser)
    text = models.TextField()
    date_joined = models.DateTimeField(default=timezone.now)
    to = models.IntegerField(blank=True)

    def __str__(self):
        return 'report: {}, user: {}, comment_id: {}'.format(self.report.title
            , self.user.username, self.id)


class ReputationLog(models.Model):
    user = models.ForeignKey(CustomUser)
    report_id = models.IntegerField(default=0)
    comment_id = models.IntegerField(default=0)
    posted = models.DateTimeField(default=timezone.now)
```

A.7 ./users/views.py

Listing 7 users アプリケーションのビュー

```
# -*- coding: utf-8 -*-

from django.views import generic
from django.contrib.auth.forms import AuthenticationForm
from django.contrib.auth import authenticate, login, logout
from django.core.urlresolvers import reverse
from django.http import HttpResponseRedirect
from django.shortcuts import render
```

```
from django.contrib.auth.mixins import LoginRequiredMixin

from .models import CustomUser
from .forms import CustomUserCreationForm


class MypageView(LoginRequiredMixin, generic.DetailView):
    login_url = '/users/signin/'
    model = CustomUser
    context_object_name = 'user'
    template_name = 'users/mypage.jinja2'


def signup(request):
    if request.method == 'POST':
        form = CustomUserCreationForm(data=request.POST)
        if form.is_valid() and form.clean_password2():
            username = request.POST['username']
            password = request.POST['password2']
            user = CustomUser.objects.create_user(username=username, password=
                password)
            if user:
                user.save()
                userauth = authenticate(username=username, password=password)
                login(request, userauth)
                return HttpResponseRedirect(reverse('users:mypage', args=(
                    userauth.id, )))
        else:
            form.errors['validerror'] = 'パスワード違うんじゃ…'
    else:
        form = CustomUserCreationForm()

    return render(request, 'users/signup.jinja2', {'form': form})


def signin(request):
    if request.method == 'POST':
        form = AuthenticationForm(data=request.POST)
        if form.is_valid():
            username = request.POST['username']
```

```
            password = request.POST['password']
            user = authenticate(username=username, password=password)
            if user is not None:
                if user.is_active:
                    login(request, user)
                    return HttpResponseRedirect(reverse('users:mypage', args=(
                        user.id, )))
    else:
        form = AuthenticationForm(request)

    return render(request, 'users/signin.jinja2', {'form': form})


def logout_view(request):
    logout(request)

    return HttpResponseRedirect(reverse('home'))
```

A.8 ./bookreports/views.py

Listing 8 bookreports アプリケーションのビュー

```
# -*- coding: utf-8 -*-

from bs4 import BeautifulSoup
from django.views import generic
from django.shortcuts import get_object_or_404, render
from django.http import HttpResponseRedirect
from django.core.urlresolvers import reverse
import requests

from .models import Book, Report, Comment, ReputationLog
from .forms import ReportForm, CommentForm

URL = 'http://www.books.or.jp/ResultList.aspx?scode=&searchtype=1&title=&
    series=&writer=&ymin=&ymax=&mmin=&mmax=&syuppansya=&isbn={}&showcount=20&
    startindex=0'
SUCCESS = 200


def book_new(request):
    if request.method == 'POST':
```

```
        registration = request.POST.get('registration')
        isbn = request.POST.get('isbn')
        name = request.POST.get('name')
        publisher = request.POST.get('publisher')
        if registration is None and isbn and Book.valid_isbn(isbn):
            r = requests.get(URL.format(isbn))

            if r.status_code == SUCCESS:
                soup_obj = Bs4Manager(r.text)
                soup_obj.isbn = isbn
                num = soup_obj.search_result_number()

                if num == 1:
                    data = soup_obj.book_data_to_dict()
        elif registration and isbn and Book.valid_isbn(isbn) and name and
            publisher:
            book = Book(id=int(isbn), name=name, publisher=publisher)
            if book:
                book.save()
                return HttpResponseRedirect(reverse('bookreports:book', args=(
                    book.id, )))
        else:
            data = {}
            data['error'] = 'を確認してくだしあ…ISBN'
    else:
        data = {}

    return render(request, 'bookreports/book_new.jinja2', {'data': data})


class Bs4Manager(object):
    def __init__(self, html_doc):
        self.soup = BeautifulSoup(html_doc, 'html.parser')

    def search_result_number(self):
        tag = self.soup.find(id='htcFoundCount')
        return int(tag.string) if tag is not None else 0

    def book_data_to_dict(self):
        table = self.soup.find('table', id='htBookList')
```

```
        # Skip to th ...
        tr = table.find_all('tr')[-1]
        keys = ['title', 'writer', 'date', 'price', 'com']
        values = list()
        for td in tr.find_all('td'):
            if td.string is None:
                tmp = list()
                for a in td.find_all('a'):
                    tmp.append(a.string)
                values.append(tmp)
            else:
                values.append(td.string)
        if len(keys) == len(values):
            data = dict(zip(keys, values))
            data['isbn'] = self.isbn
            return data
        else:
            None


class BooksView(generic.ListView):
    template_name = 'bookreports/books.jinja2'
    context_object_name = 'books'

    def get_queryset(self):
        return Book.objects.all()


class BookView(generic.DetailView):
    model = Book
    context_object_name = 'book'
    template_name = 'bookreports/book.jinja2'


def report_new(request, pk):
    if request.method == 'POST':
        form = ReportForm(request.POST)
        if form.is_valid():
            title = request.POST['title']
            text = request.POST['text']
```

```
            user = request.user
            book = Book.objects.get(pk=pk)
            report = Report(user=user, book=book, title=title, text=text)
            if report:
                report.save()

                user = report.user
                user.reputation += 2
                user.save()
                return HttpResponseRedirect(reverse('bookreports:report', args
                    =(pk, report.id)))
    else:
        form = ReportForm()
    return render(request, 'bookreports/report_new.jinja2', {'form': form})


def reports(request, pk):
    book = get_object_or_404(Book, pk=pk)
    reports = Report.objects.filter(book=book)

    return render(request, 'bookreports/reports.jinja2', {'reports': reports,
        'book': book})


def report(request, book_pk, report_pk):
    report = get_object_or_404(Report, pk=report_pk)

    if request.method == 'POST':
        form = CommentForm(request.POST)
        if form.is_valid():
            to = int(request.POST['to'])
            text = request.POST['text']
            user = request.user
            comment = Comment(report=report, user=user, text=text, to=to)
            if comment:
                comment.save()
                return HttpResponseRedirect(reverse('bookreports:report', args
                    =(book_pk, report_pk)))
    else:
        form = CommentForm()
```

```
    comments = Comment.objects.filter(report=report).order_by('date_joined')

    context = {
        'report': report, 'form': form, 'comments': comments
    }

    return render(request, 'bookreports/report.jinja2', context)


def reputation(request, book_pk, report_pk):
    user = request.user
    comment_id = request.POST.get('comment')
    if comment_id is None:
        report = get_object_or_404(Report, pk=report_pk)
        user = report.user
        user.reputation += 5
        user.save()

        log = ReputationLog(user=user, report_id=report.id, comment_id=0)
    else:
        comment = get_object_or_404(Comment, pk=int(comment_id))
        user = comment.user
        user.reputation += 10
        user.save()

        log = ReputationLog(user=user, report_id=0, comment_id=comment.id)
    log.save()
    return HttpResponseRedirect(reverse('bookreports:report', args=(book_pk,
        report_pk)))
```