筑波大学大学院博士課程

システム情報工学研究科特定課題研究報告書

問題解決の実践によるプログラミング教育向け学習支援システム

—Web アプリケーションの開発と Web サイトからのソースコード剽窃チェック機能の実装—

林 東火

修士（工学）

(コンピュータサイエンス専攻)

指導教員 田中 二郎

２０１６年 ３月

概要

今日の大学のプログラミング教育のやり方では，アルゴリズムの勉強では，理論を学ぶことが多くプログラムを書く機会が少ない．学生にプログラムを書くチャンスを増やすため，筑波大学の「プログラミングチャレンジ」授業ではプログラミングコンテストの問題を利用してプログラムを書いて学習している．しかし，利用している外部サービスは教育用に向いていないため，様々な問題が存在している．

これらの問題を解決するため，筆者のチームはプログラミング教育用を前提として，授業管理機能，自動評価機能，解答ソースコードの剽窃チェック機能付きのオンラインサービスの開発を行った．筆者は Web アプリケーションの開発と Web サイトからのソースコード剽窃チェック機能の開発を担当した．Web サイトからのソースコード剽窃チェック機能では，検索エンジンを利用して，Web サイトから関連ページを検出することである．実装した剽窃チェック機能について，評価実験を行った．実験の結果から，Web サイトからのソースコード剽窃チェック機能の有効性を証明した．

目次

図目次

表目次

第1章　はじめに

本報告書は，筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻高度 IT 人材育成のための実践的ソフトウェア開発専修プログラム（以下，高度 IT コースと記載）における特定課題研究「問題解決の実践によるプログラミング教育向け学習システム」において，2015 年度に実施したプロジェクトについてまとめたものである．

本報告書では，「本プロジェクト」は「問題解決の実践によるプログラミング教育向け学習システム」を表す．プロジェクトタイトルの「問題」はプログラミング問題である．本報告書では，利用している画像には，「問題」と言う単語が存在する．指定がなければ，「問題」はプログラミング問題を表す．

本プロジェクトの顧客は筑波大学の「プログラミングチャレンジ」科目担当の先生である．プロジェクトのメンバは高度 IT コース M2 の 3 名学生である．指導教員は筑波大学大学院コンピュータサイエンスに所属する教員である．

本プロジェクトは，アジャイル開発手法でプロジェクトを推進した．筆者のチームは顧客が抱えている課題を解決すると目指し，課題の分析，ソリューションの提案，システムの構築，実験検証などを行った．

本プロジェクトで開発したシステムは「ログイン機能」，「授業管理機能」，「問題管理機能」，「解答評価機能」，「解答管理機能」，「解答差分表示機能」，「ユーザ管理機能」，「剽窃チェック機能」などの機能がある．開発したシステムには，「学生」，「教員」，「管理者」，「システム管理者」4 種類のユーザがある．

剽窃チェック機能は学生間解答をコピーしたかどうかの剽窃チェック（以下，ローカル剽窃チェックと記載）と Web サイトからそのままコピーしたかどうかの剽窃チェック（以下，Web 剽窃チェックと記載）がある．ローカル剽窃チェック機能では，指定されたソースコードの間の共通点を探し，ソースコードの間の類似度を計算することである．Web 剽窃チェック機能では，検索エンジンを利用し，Web サイトから関連ページを検出することである．

本報告書は本章を含めて全 6 章と付録で構成されている．第 2 章では開発背景とプロジェクト概要につて述べる．第 3，4 章では Web アプリケーションの開発と Web サイトからのソースコードの剽窃チェック機能の開発について述べる．第 5 章では評価実験について述べる．第 6 章ではまとめと今後の課題について述べる．また，付録として開発したアプリケーションの仕様書を添付する．

第2章　開発背景とプロジェクト概要

本章では，本プロジェクトの背景，概要について紹介する．開発背景では顧客が抱える問題も述べる．

2.1　背景

近年，世界中プログラミング教育が盛んになってきている．多くの国で小学校からプログラミング教育が実施されている．日本も「義務教育段階からのプログラミング教育等の IT 教育を推進する」ということを議論している．今後，学校のプログラミング教育は益々重要になる．一方，レベルが高い大学のプログラミング教育のやり方では，アルゴリズムの勉強では，理論を学ぶことが多くプログラムを書く機会が少ない傾向がある．

学生にプログラムを書くチャンスを増やすため，筑波大学の「プログラミングチャレンジ」授業ではプログラミングコンテストの問題を利用してプログラムを書いて学習している．プログラミング問題の解決を通じて，学生のプログラムを書く技術を高めること以外，アルゴリズムの活用，プログラミング問題解決技法を学ぶこと，問題解決の思考力を鍛えることなど様々な面で効果がある．

2.2　顧客の現状と要望

2.2.1　顧客の現状

顧客は現在授業で使用するプログラミング問題およびプログラムの評価に外部の Web サービス Programming Challenges [1]，UVa Online judge [2]と筑波大学校内の学習管理システム Manaba[3]など複数のサービスを利用している．利用している外部の Web サービスは教育用に向いていないため，図 2-1 に示すように，4 つの問題が存在している．

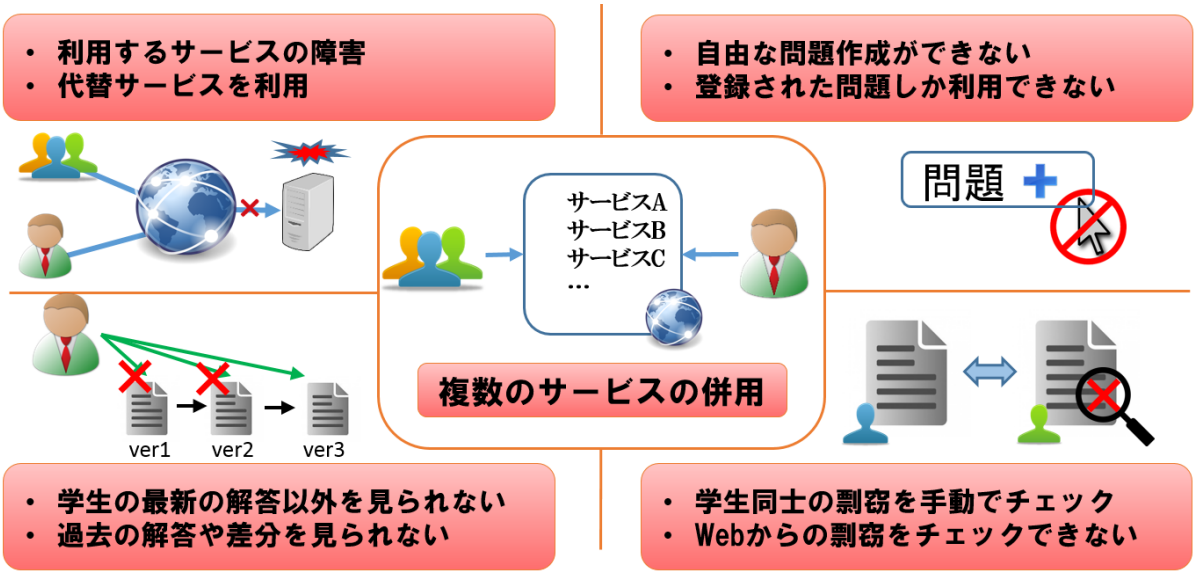

図 2-1　現状の問題点

問題 1, 顧客は外部の Web サービスを 3 年間使っているが, サーバのダウンが毎年発生し,

授業に参加する学生は課題の解答が出来なくなることがある．また，サーバのダウンの間，顧客は別のサービスを利用して，学生に課題を与える．

問題 2，顧客が使用している外部のサービスはプログラミング問題を作る権利がない．顧客は，外部の Web サービスに登録しているプログラミング問題を選んで，学生に与える．しかし，プログラミング問題を作る権利がないため，学生に適切な，プログラミング問題を出せない場合がある．

問題 3，顧客が使用している外部のサービスには学生の解答履歴を閲覧できない．解答の差分機能がない．1 つのプログラミング問題について，提出の締切日まで，何回も提出できるため，解答履歴には，多数の解答がある．解答履歴を閲覧できないため，学生の解答の状況が分かりにくい，学生に適切な指導を与えられない．

問題 4，顧客が使用している外部のサービスは剽窃チェック機能がない．1 つの授業に多数の学生がいる，学生同士がお互いコピーを行っているかどうか調べるのは時間がかかる．また外部の Web サービスのプログラミング問題を利用するので，プログラミング問題の解答は既に Web サイトで公開している可能性があるため，学生の解答は Web サイトからコピーしたかどうか知ることができない．

2.2.2 顧客の要望

顧客は教育を目的として，学生が提出したプログラムを自動評価するシステムを開発することを要求している．具体的には下の 4 つの要望がある．

要望 1，ユーザ管理機能，プログラミング問題管理機能，授業管理機能の作成．

要望 2，学生の提出したプログラムをサンドボックス環境で自動評価する機能の作成．

要望 3，解答履歴を閲覧できる，各回提出したプログラムの差分表示機能の作成．

要望 4，提出されたプログラムのソースコードの剽窃チェック機能の作成．

2.3 既存システムの調査

筆者のチームは顧客の要望に基づいて，顧客利用している校内学習管理システム Manaba，各大学公開しているサービス，国内外オンラインジャッジサービスについて調査した．調査した結果を表 2-1 に示す．日本の大学が公開しているプログラミング教育用の Web 電子管理システムがあまり見つからない．一方，プログラミングコンテストサイトやオンラインジャージサイトがいくつ存在している．また，企業経営しているオンラインプログラミング学習サイトもたくさん存在している．

表 2-1　既存システムの比較

	学生管理	クラス，コンテスト作成	プログラミング問題作成	自動評価	学生の解答履歴の閲覧	差分表示	剽窃チェック	日本語対応
Manaba	○	○	△	×	×	×	×	○
Programming Challenges	○	○	×	○	○	×	×	×
Moodle[4]	○	○	○	○	○	×	×	○
UVa Online judge	×	×	×	○	○	×	×	×
AIZU ONLINE JUDGE[5]	×	×	×	○	○	×	×	○
東京大学プログラミングコンテスト[6]	×	×	×	○	○	×	×	○

Manaba は学習管理システムである．プログラミング問題の作成は可能だが，自動評価機能，履歴閲覧機能，差分表示機能，剽窃チェック機能がない．

Programming Challenges は海外のオンラインジャージサービスである．学生管理機能，クラスの作成機能，自動評価機能，解答履歴の閲覧機能などが利用できるが，問題作成，差分表示，剽窃チェック機能がない．

Moodle は学習支援システムである．学生管理機能，クラスの作成機能，問題作成機能，自動評価機能，解答履歴の閲覧機能などが利用できるが，差分表示機能，剽窃チェック機能がない．

UVa Online judge は海外のオンラインジャージサービスである．学生管理機能，クラスの作成機能，問題作成機能，差分表示機能，剽窃チェック機能などが提供していない．

AIZU ONLINE JUDGE は会津大学提供しているオンラインジャージサービスである．日本語と英語を対応している．利用者は個人である．学生管理機能，クラスの作成機能，問題作成機能，差分表示機能，剽窃チェック機能などが提供していない．

東京大学プログラミングコンテストは東京大学が提供しているオンラインジャージである，学生管理機能，クラスの作成機能，問題作成機能，差分表示機能，剽窃チェック機能が提供していない．

2.4　ソリューション

先述の顧客の現状及び顧客からの要望をうけ，筆者のチームは図 2-2 のような Web 電子管理システムを顧客に提案した．

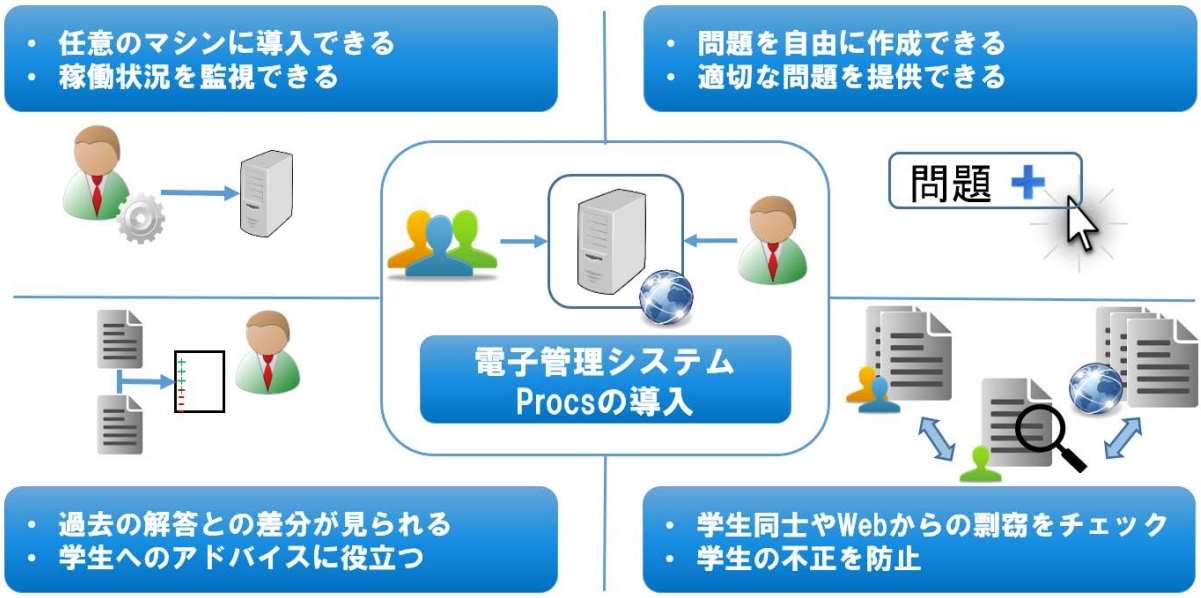

図 2-2　提案システム

提案した Web 電子管理システムは 6 つ機能がある．1，授業管理機能．2，プログラミング問題管理機能．3，ユーザ管理機能．4，学生の提出したプログラムをサンドボックス環境で自動評価する機能．5，各回提出したプログラムの差分表示機能．6，提出された解答ソースコードの剽窃チェック機能．

2.5　　システムの概要

本プロジェクトで開発した電子管理システムでは, ユーザ管理, プログラミング問題管理, 授業の作成，自動評価，解答履歴の閲覧，履歴の差分表示など機能がある．自動評価では C, C++,Python 言語をサポートしている．また，教育用に向いているため，ローカル剽窃チェック機能と Web 剽窃チェック機能が開発した．

2.5.1　　システム構成

本システムの構成を図 2-3 に示す．利用者は Web アプリケーションで，プログラミング問題の解答を提出する．バックエンドでは，提出した解答は実行待ちキューに入れ，実行タイムが来たら，Docker[7] サンドボックス環境で解答プログラムは全てのテストデータを実行する．実行の結果はデータベースに保存し，Web アプリケーションで利用者に結果を表示する．

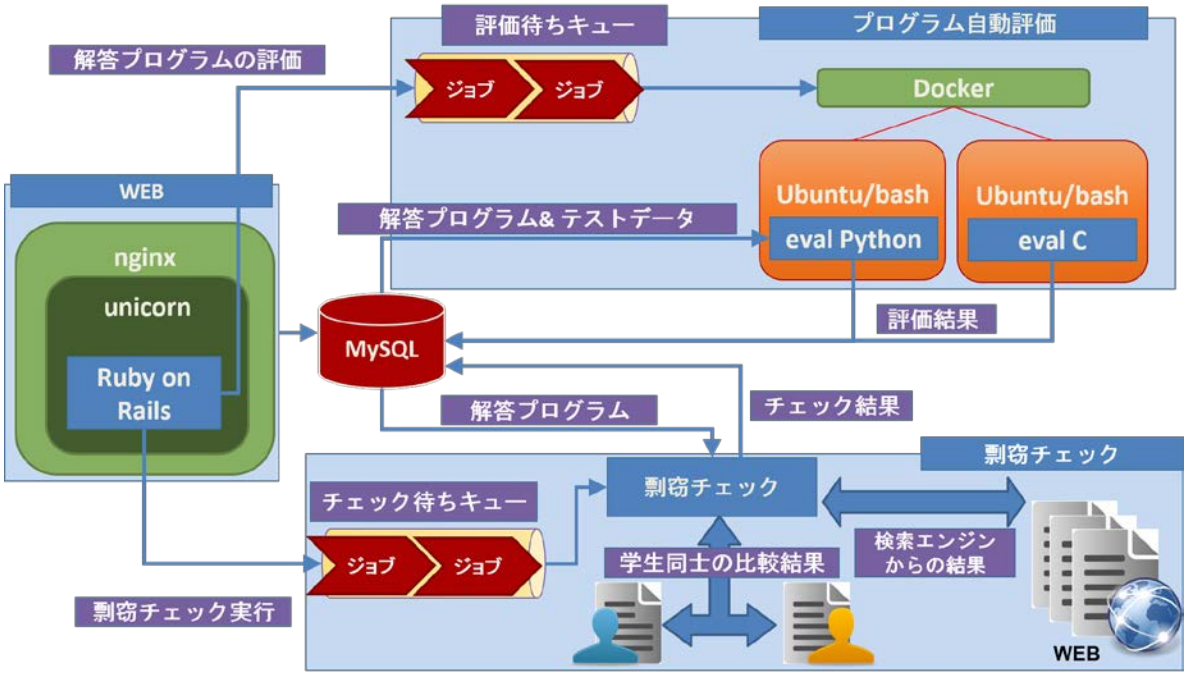

図 2-3　システム構成図

また，評価が Accept の解答は自動的にローカル剽窃チェックが行われる．利用者は教員の場合，授業に登録したプログラミング問題の解答について， Web 剽窃チェックが行える．剽窃チェックの結果はデータベースに保存する．剽窃チェックの結果は教員に表示する．

2.5.2　　システム機能

本システムの利用者は学校の教員，学生を想定している．開発したシステムは以下７つの機能がある．

（1）ログイン機能

ログイン機能には，登録機能，認証機能がある．

（2）ユーザ管理機能

ユーザ管理機能には，ユーザの新規作成，編集，削除機能がある．

（3）授業管理機能

授業管理機能には，授業の新規作成，編集，削除機能と授業参加する機能がある．

（4）プログラミング問題管理機能

プログラミング問題管理機能では，プログラミング問題の新規作成，編集，削除機能がある．

（5）自動評価機能

自動評価機能は，提出した解答ソースコードに対して，プログラミング問題テストデータを利用して評価する．

（6）解答管理機能

解答管理機能には，解答の作成，削除機能，解答の差分表示する機能がある．

（7）剽窃チェック機能

剽窃チェック機能には，ローカル剽窃と Web 剽窃チェック機能がある．

2.6 開発環境

本システムの開発環境及び利用したツールを表 2-2 に示す.

表 2-2 開発環境及び利用ツール

項目	名称	バージョン
OS	Ubuntu	14.04 64bit
開発言語	Ruby	2.2.3
フレームワーク	Ruby On Rails	4.2.1
Web サーバ	Nginx[8]	1.4.6
アプリサーバ	Unicorn[9]	5.0.1
データベース	Mysql	5.6.19
サンドボックス	Docker	1.8.2
ローカル剽窃チェックツール C/C++	Sim[10]	2.70
ローカル剽窃チェックツール Python	CloneDigger[11]	1.1.0
検索エンジン	Bing[12]	
プロジェクトの管理ツール	Taiga[13]	
バージョン管理ツール・システム	Git，Github	

Ruby On Rails は生産性が高いため, 今回の開発フレームワークは Ruby On Rails にした. データベースは Mysql にした. Unicorn は高速なクライアントとレスポンスできる特徴があるため，アプリケーションサーバは Unicorn にした．また Nginx メモリ消費が少ない, Unicorn へのサポートが良いため，Web サーバは Nginx にした.

Docker は Linux コンテナの中でアプリケーションを動かすためのオープンソースツールである．配置が簡単かつ軽量的な特徴があるため，本プロジェクトでは，Docker を利用してサンドボックス環境を構築した.

SIM は C/C++, Java, Pascal, Modula-2, Miranda, Lisp 向けのソースコードテキストの類似性テスターである．Clonedigger は Python で開発した Python，Java 向けのクローン検出ツールである．本プロジェクトでは SIM , Clonedigger を利用してローカル剽窃チェック機能を実装した.

Bing は Microsoft の検索エンジンである．本システムでは，Bing 検索の API を利用し Web 剽窃チェック機能を実装した.

2.7 プロジェクトの進め方

本プロジェクトは 2 週間 1 スプリントとし，全部 2 つのイテレーションで開発を行った. 第 1 イテレーションでは，Web アプリケーションのプロトタイプを作成した．第 2 イテレーションでは，サンドボックス環境で自動評価する機能，ソースコードの剽窃チェック機能の実装を行った.

2.7.1 プロジェクトの管理

筆者のチームは，Taiga を利用してプロジェクトを管理した．Taiga は，本格的なアジャイル開発を支援するプロジェクト管理ツールであり，オープンソースで自分のホストへのインストールの場合は無料になる．図 2-4 は Taiga の KANBAN を示している.

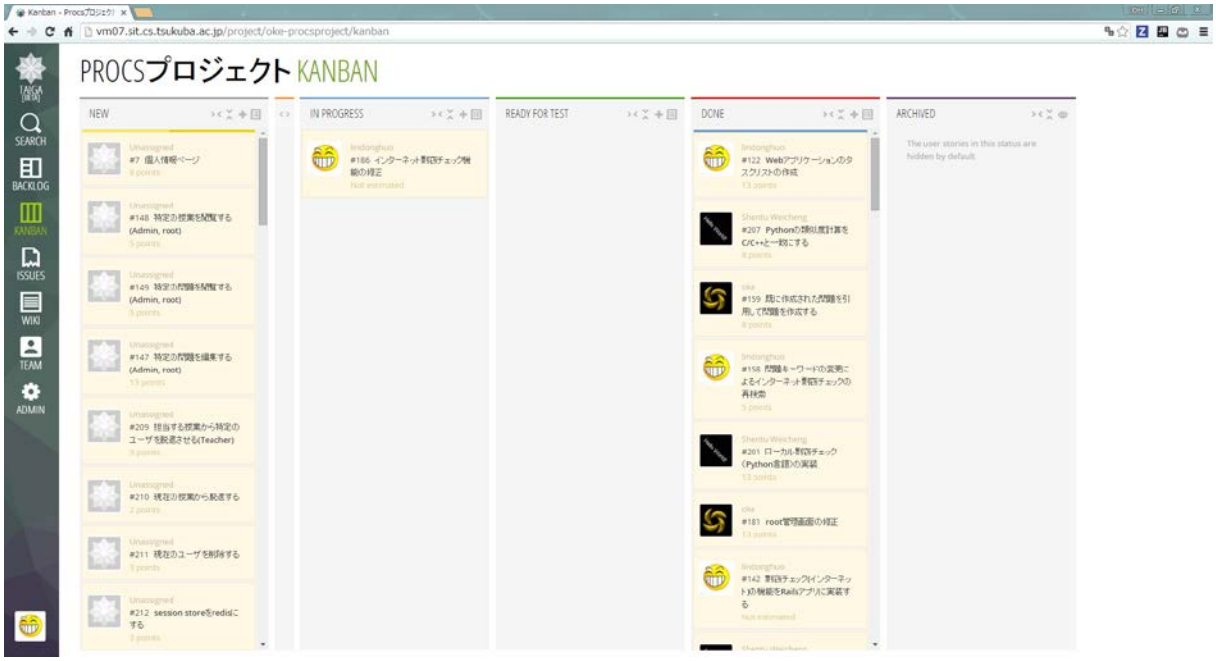

図 2-4　Taiga KANBAN 画面

本プロジェクトでは，チームメンバ共同でシステムの開発を行ったため，タスクの管理はなるべく簡単化，可視化にしたい．Taiga の KANBN は，プロジェクトマネジメントで速やかに同期できる特徴があるため，複数人で共同作業の場合，KANBN は使いやすい．また，Taiga にはタスク見積機能，スプリント管理機能があり，ほかには ISSUES，WIKI など機能もあるため，プロジェクトの管理は Taiga にした．

2.7.2　　開発作業の分担

本プロジェクトの開発チームは 3 人のメンバから構成される．本プロジェクトにおいて，実際の各段階の作業の担当を表 2-3 に示す．要求定義，詳細設計の作業はチームメンバ全員共同で作業した．第 1 イテレーションの開発では， Web アプリケーションの「学生，教員共通のマイページ」，「問題新規ページ」，「ヘッダの作成」，「タブバーの作成」など機能を実装した． 第 2 イテレーションの開発では， Web アプリケーションの問題管理，授業管理機能の開発と Web 剽窃チェック機能の開発を担当した．各担当作業の詳しい説明は第 3 章と第 4 章で紹介する．

表 2-3　開発作業の分担

工程	作業内容	担当者
要求定義	顧客の既存問題の調査 既存システムの調査 提案	大桶，林，申屠
詳細設計	データベースの設計 画面設計	大桶，林，申屠
第 1 イテレーションの開発	環境構築 Web アプリケーションのプロトタイプの開発	大桶，林，申屠
第 2 イテレーションの開発	サンドボックス環境の構築 Web アプリケーションの管理者の機能	大桶
	ローカル剽窃チェック	申屠
	Web 剽窃チェック Web アプリケーションの問題管理，授業管理	林

2.7.3　　開発スケジュール

本プロジェクトを開始した時に作成したスケジュールを図 2-5 に示す．全部 3 イテレーションでプロジェクトを推進する予定があった. 第 1 イテレーションでは, 5 月から 8 月まで, 要件定義，詳細設計，プロトタイプの作成を行う予定があった．第 2 イテレーションでは, 9 月から 10 月まで, サンドボックス環境の構築を行う予定があった. 第 3 イテレーションでは，11 月から 12 月まで，剽窃チェックチェックの実装を行う予定があった． 8 月の月末, 10 月の月末，12 月の月末に納品する予定があった.

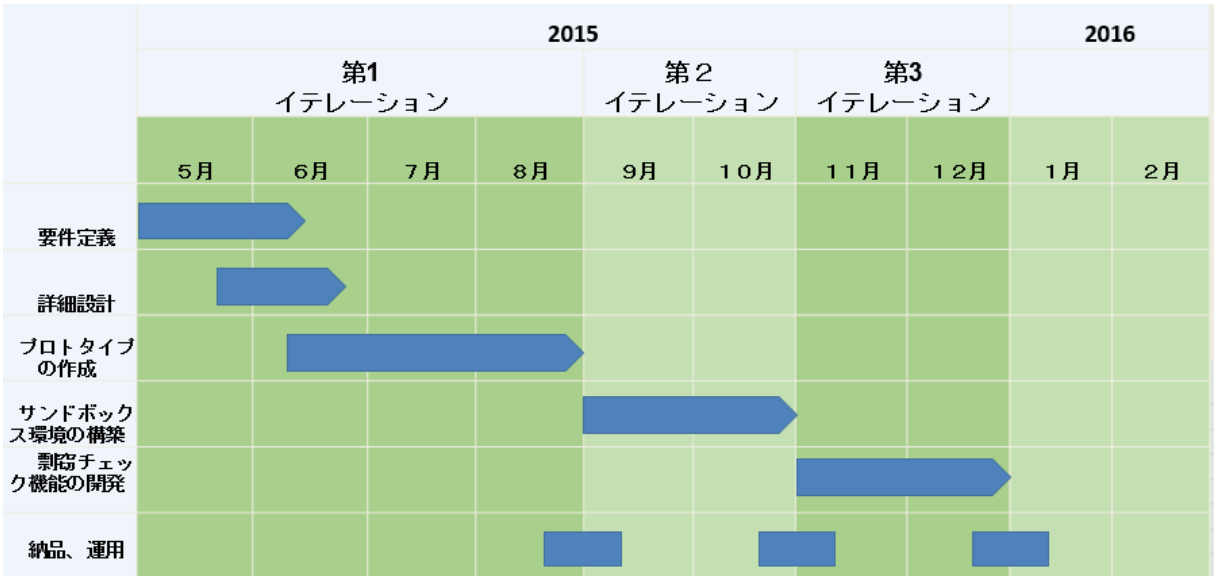

図 2-5　修正前のスケジュール

しかし，7 月の中間報告で，第 3 イテレーションの剽窃チェック機能は 11 月から開発すると，プロジェクトが失敗になる恐れがあると指摘があった．そして，予定のスケジュールを修正した．修正後のスケジュールを図 2-6 に示す．青いラインが修正後の予定のスケジュールである．最初全部で 3 イテレーションから全部で 2 イテレーションに変更した．第 2 イテレーションから，サンドボックス環境の構築，剽窃チェック機能の開発，Web アプリケーションの修正を同時に推進する予定があった．

図 2-6 中の赤いラインが実際のスケジュールである．第 1 イテレーションは予定どおりで進んだ．第 2 イテレーションは予定よりやや遅れた．最終の納品は 12 月の月末に行った．

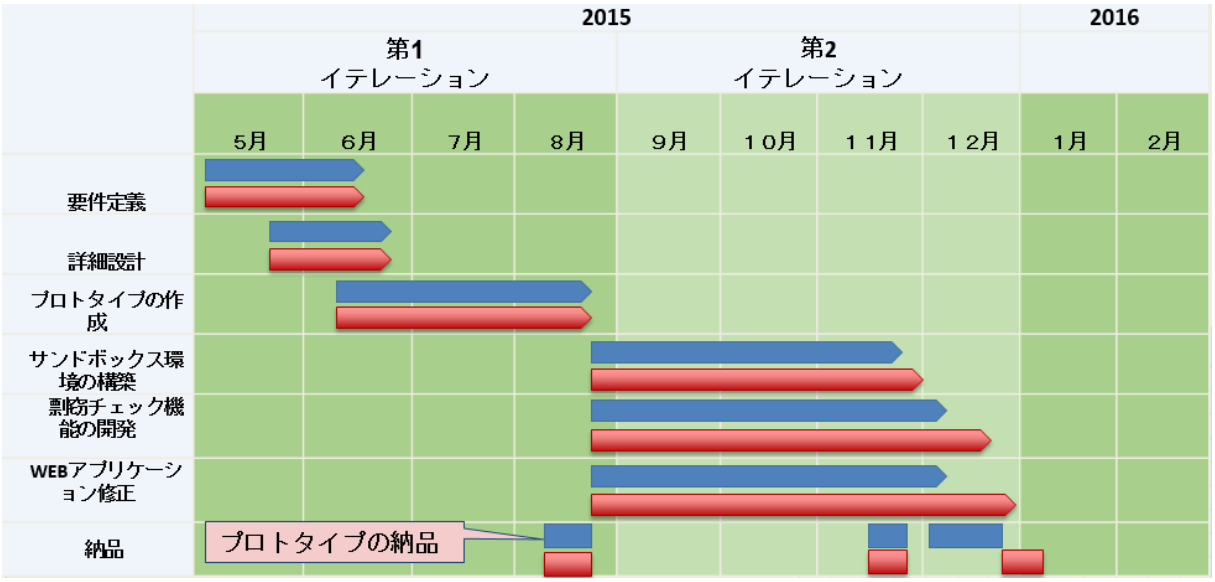

図 2-6　修正後のスケジュール

第3章　Web アプリケーションの開発

本プロジェクトは Ruby On Rails で Web アプリケーションを開発した．開発した Web アプリケーションはユーザ管理, 授業管理, プログラミング問題管理, プログラムの自動評価, 学生の評価管理など機能がある．本章では，筆者は Web アプリケーション開発で担当した作業について紹介する．

3.1　Ajax による画面遷移

本システムの授業画面を設計した時，筆者のチームは Ajax による画面遷移を行うように設計した．Ajax とは，ブラウザ上デスクトップアプリケーションのようなリッチな表現を可能にする技術のことである．作成した授業画面を図 3-1 に示す．授業画面では，タブバーを利用する．タブバーには「授業」,「問題」,「学生」3 つのタブがある．タブバーを切り替えする時，URL（図 3-1 の赤い枠）は変更せず，タブバー下のエリア（図 3-1 の青い枠）の中身だけが変わる．

Ajax による画面遷移を行うように設計した理由としては 2 つがある．1 つ目は，システムのセキュリティを考えた．Ajax を利用して，URL 変更せずに授業画面を作成した． URL には色々な情報がある．Ajax を利用すると，ユーザに表示する情報を減らした．また，URL からの非 Ajax のアクセスを制限したため，システムのセキュリティを強化した．2 つ目は，ユーザの利便性を考えた．Ajax を利用しない場合，操作によって，URL を変更する．URL を変更すると，ページ全体の更新が行われる．ページ全体更新によるーデータのやり取りを頻繁に行い，ページを表示するスピードが落ちる．Ajax を利用する場合，一部エリアだけを更新する．本システムの授業画面は，ページ全体の更新が行われない，タブバー下のエリアだけを更新する．ページの表示スピードが上がる．

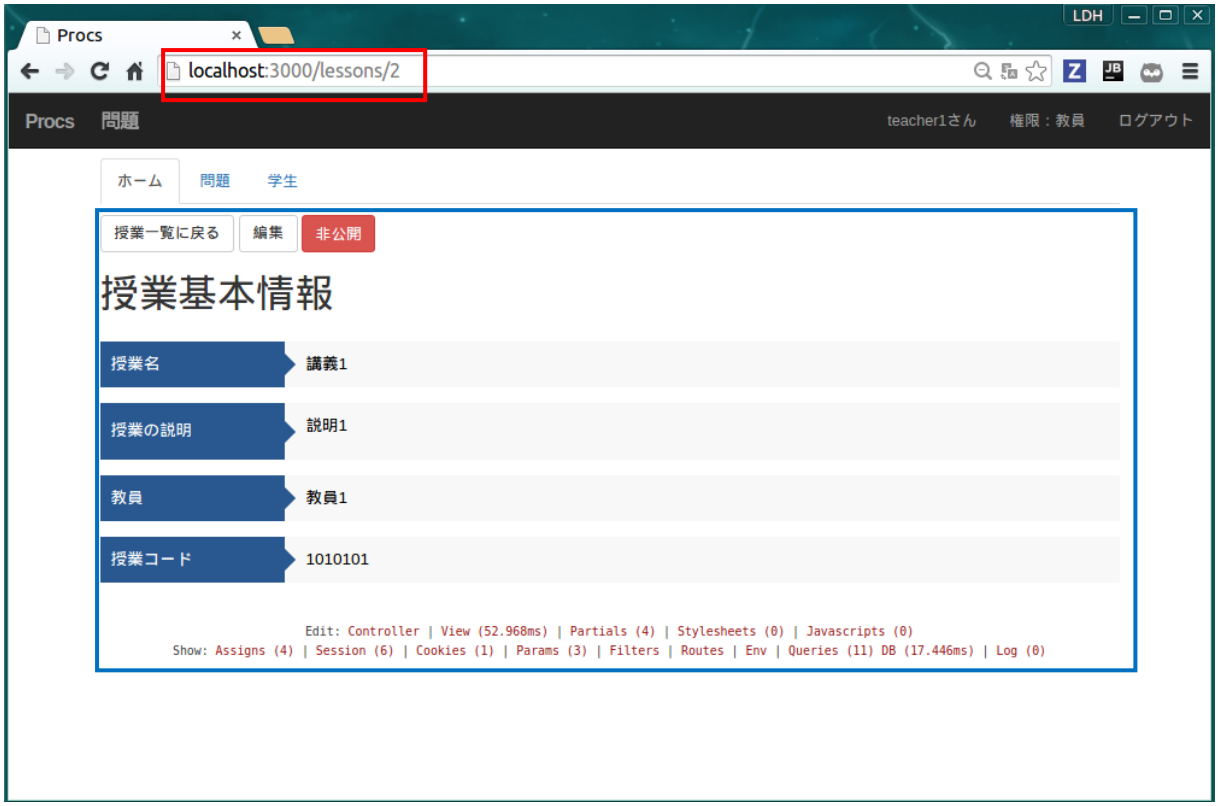

図 3-1　授業画面

本システムは Ruby On Rails で開発したものである．Rails では，link_to，form_tag，form_for などのビューヘルパーが Ajax 通信のための機能を提供している．筆者は JavaScript に詳しくないため，Rails の Ajax 通信機能を利用し，画面遷移を実装した．Rails の Ajax 通信機能は使用しやすいため，実装の工数も削除した．

3.2　プログラミング問題の作成

プログラミング問題作成機能の機能要件を表 3-1 に示す．プログラミング問題の作成は授業の担当教員である．プログラミング問題には，タイトル，内容，入力説明，出力説明，開始時間，終了時間，Web 剽窃チェック用のキーワード，サンプルデータ，テストデータ，制限実行時間，利用メモリなどデータがある．Web 剽窃チェック用のキーワードとテストデータは教員にのみ表示する．Web 剽窃チェック用のキーワード，サンプルデータ，テストデータを追加，削除できる．大きいサイズのテストデータがあるため，テストデータはファイルの形でアップロードする．また，プログラミング問題は授業中だけでなく，全員が利用できるように設定もできる．

表 3-1　プログラミング問題作成機能の機能要件

機能要件
プログラミング問題は授業内だけでなく，全員が利用できるように設定ができる(パブリック問題)
自分の作成したプログラミング問題，またはパブリック問題を引用して作成することができる
プログラミング問題の作成はその授業の担当教員のみである
プログラミング問題のテストデータはファイルの形でアップロードする
プログラミング問題のサンプルデータはテキストの形で入力する
Web 剽窃チェック用のキーワードは追加，削除できる
プログラミング問題の開始時間と終了時間を設定できる
プログラミング問題のサンプルデータを追加，削除できる
プログラミング問題のテストデータを追加，削除できる
プログラミング問題の解答の実行時間を入力する
プログラミング問題の解答の使用メモリを入力する
プログラミング問題のテストデータは担当教員のみを表示する
Web 剽窃チェック用のキーワードは担当教員のみを表示する
プログラミング問題のタイトル，内容，入力説明，出力説明を入力する

プログラミング問題を新規作成するとき，新規作成する以外に，パブリック問題から引用して作成することもできる．作成したプログラミング問題作成画面を図 3-2 に示す．

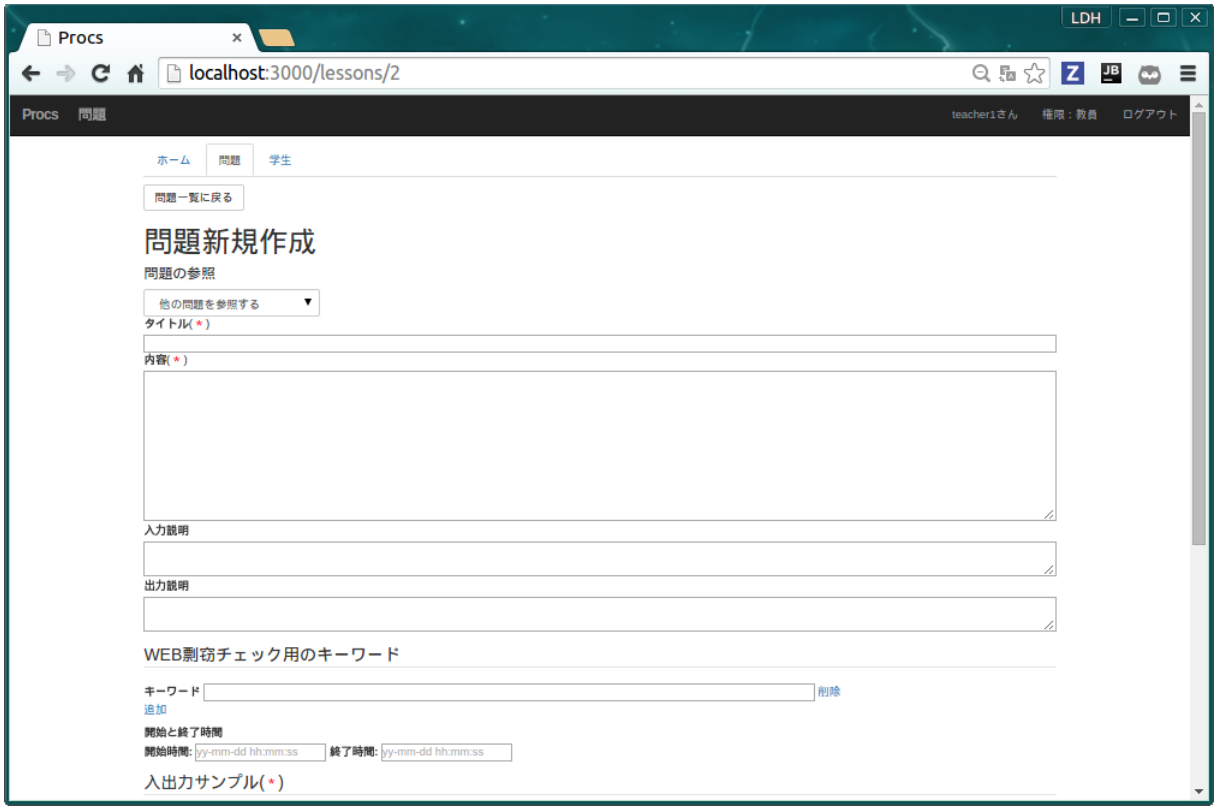

図 3-2　プログラミング問題作成画面

筆者はプログラミング問題作成の機能を実装した時, ライブラリ nested_form を利用した. プログラミング問題のテーブル questions では，多数のテーブルと関連している．関連している要素の追加・削除では，実装のコストがかかる．ライブラリ nested_form は，1 体多，多対多の関係があるフォームを作る時，関係フォームの追加，削除するとき，非常に便利である．本システムでは，nested_form を利用し，Web 剽窃チェック用のキーワード，サンプルデータ，テストデータなど要素を動的に追加・削除できるように実装した.

3.3　プログラミング問題の編集

プログラミング問題編集機能の機能要件を表 3-2 に示す．プログラミング問題の編集は教員のみである．教員は自分作成した問題を編集できるが，パブリック問題からコピーして作成したプログラミング問題については編集できない. また, プログラミング問題のタイトル，内容，入力説明，出力説明，開始時間，終了時間，Web 剽窃チェック用のキーワード，サンプルデータ，テストデータ，実行時間，使用メモリについて編集できる．また，教員自分作成したプログラミング問題，一度パブリック問題に公開した後，非公開に編集できない.

表 3-2　プログラミング問題編集の機能要件

機能要件
プログラミング問題の編集は教員のみである
教員は自分作成したプログラミング問題について編成できる
教員は自分作成した非公開プログラミング問題はパブリック問題に編集できる
教員は自分作成したプログラミング問題，一度パブリック問題に公開した後，非公開に編集できない
プログラミング問題の編集はその授業の担当教員のみである
Web 剽窃チェック用のキーワードを編集できる
プログラミング問題の開始時間と終了時間を編集できる
プログラミング問題のサンプルデータを編集できる
プログラミング問題のテストデータは追加，削除できる
プログラミング問題の解答の実行時間の制限を編集できる
プログラミング問題の解答の使用メモリの制限を編集できる
プログラミング問題のタイトル，内容，入力説明，出力説明を編集できる

筆者はプログラミング問題編集の機能を実装した時，プログラミング問題を編集する前，既に正解の解答があったと言うケースについて検討した.プログラミング問題を編集した後，これらの解答はもう 1 回評価する必要がある．この問題の解決方法は 2 つがある．1 つ目はプログラミング問題編集した後，提出した解答は自動的に再評価する．2 つ目は学生にもう 1 回解答を提出すると言う情報を提示する．教育の視点から見ると，学生がプログラミング問題を確認してから，もう 1 回解答を提出するのは合理である．本システムでは，プログラミング問題を編集した後，Accept になった解答について，自動的に再評価せず，該当学生に再提出のメッセージを提示する．

3.4　プログラミング問題の削除

プログラミング問題削除機能の機能要件を表 3-3 に示す．プログラミング問題の削除は教員のみである．教員は自分作成したプログラミング問題を削除できる．しかし，パブリック問題からコピーした問題は，削除の代わりに，非公開にする．自分作成したプログラミング問題はパブリック問題に公開した後，パブリック問題からコピーした問題と同じ，削除の代わりに，非公開にする．また，プログラミング問題の解答が存在する場合，削除の代わりに，非公開になる．一度非公開したプログラミング問題は，授業に参加している学生が閲覧できないが，担当教員が閲覧できる．管理者は非公開プログラミング問題を再公開できる．

表 3-3　プログラミング問題削除の機能要件

機能要件
プログラミング問題の削除はその授業の担当教員のみである
パブリック問題は，削除の代わりに，非公開にする
プログラミング問題の解答が存在する場合，削除の代わりに，非公開にする
一度非公開にしたプログラミング問題は，管理者が再公開できる
担当教員は非公開にしたプログラミング問題を閲覧できる

プログラミング問題削除機能の実装では，実際のデータの削除が行われない．問題と授業を関連しているテーブルに Flag を追加する．削除の場合，Flag の値は True にする，非削除の場合は，Flag の値は False にする．Flag の値によって，プログラミング問題を表示・非表示する．実際のデータの削除は管理者が行われる．

3.5　学生一覧ページの作成

学生一覧ページの機能要件を表 3-4 に示す．学生一覧は教員のみ表示する，学生権限のユーザに表示しない．学生一覧ページでは授業に参加している学生表示する．

表 3-4　学生一覧ページの機能要件

機能要件
学生一覧は教員のみ表示する
学生の一覧を表示する
学生の学籍番号，ニックネーム，成績の情報を表示する
学生詳細ページにアクセスできる

実装した学生一覧ページを図 3-3 に示す．授業に参加している学生はリストで表示する．本システムでは，学生一覧ページでは学生の学籍番号，ニックネーム，成績だけを表示する．各学生の学籍番号にクリックすると，該当学生の詳細ページにアクセスできる．成績は Accept になっている解答の割合を表示する．学生一覧ページはシンプルであるが，今後，他の表示したいデータがあれば，まだ追加する．

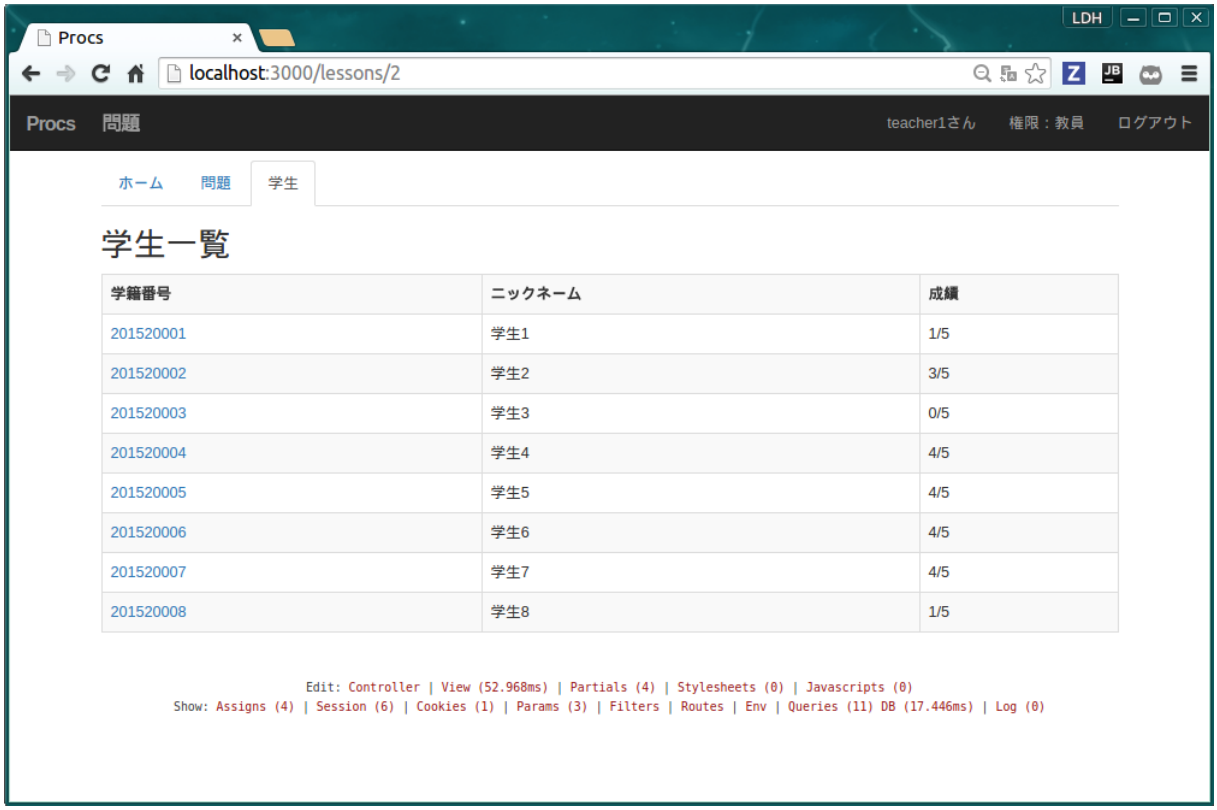

図 3-3　学生一覧ページの画面

3.6　　学生詳細ページの作成

学生一覧ページの機能要件を表 3-5 に示す．学生詳細ページは教員のみ表示する，学生権限のユーザに表示しない．学生詳細ページでは学生の名前，学籍番号，email など個人情報と各プログラミング問題の解答状況を表示する．

表 3-5　学生詳細ページの機能要件

機能要件
学生詳細は教員のみ表示する
学生の名前，学籍番号，email アドレスの情報を表示する
プログラミング問題ごとの解答状況を表示する
Web 剽窃チェックを利用できる
解答詳細ページにアクセスできる

実装した学生詳細ページを図 3-4 に示す．個人データと授業内全てのプログラミング問題の解答状況を表示する．解答状況リストで，各問題の解答状況を表示する．プログラミング問題にクリックすると，該当問題の解答詳細ページにアクセスできる．また，「内容を確認」をクリックしたら，ローカル剽窃チェックの結果はこのページで確認できる． 一方，Accept になった解答は，「チェック」をクリックすると，Web 剽窃チェックが行われる．既にチェック済みの解答は「表示」をクリックすると，結果を表示する．

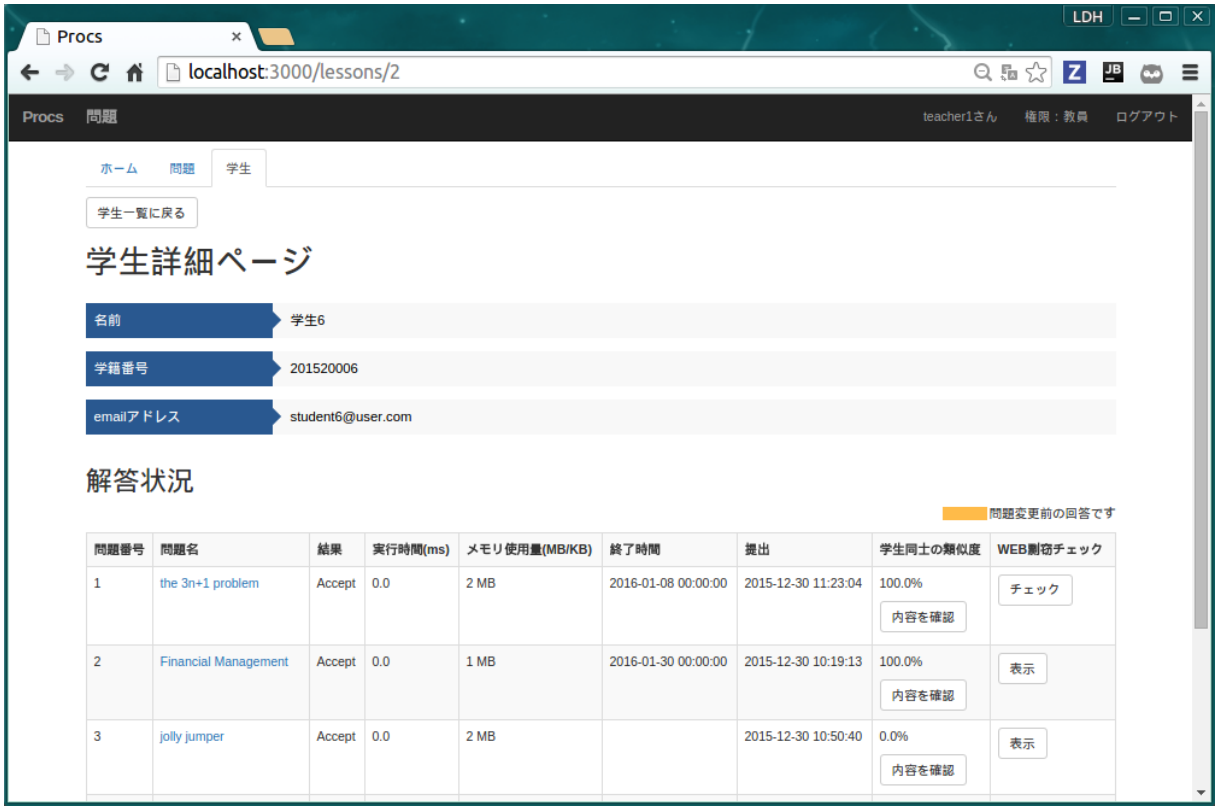

図 3-4　学生詳細ページの画面

3.7　授業編集

授業編集機能の機能要件を表 3-6 に示す．授業編集は教員のみである．教員は自分作成した授業を編集できる．本システムで開発した授業編集機能は授業名，授業の説明だけ編集できる．授業コード，担当教員のデータを編集できない．

表 3-6　授業編集機能の機能要件

機能要件
授業の編集はその授業の担当教員のみである
授業コード，担当教員は編集できない
授業名，授業の説明は編集できる

3.8　授業削除

授業削除機能の機能要件を表 3-7 に示す．授業削除は教員のみである．教員は自分作成した授業を削除できる．授業にプログラミング問題が存在する場合，削除の代わりに，非公開になる．一度非公開した授業は，担当教員が閲覧できるが，授業に参加している学生が閲覧できない．非公開授業を再公開したい場合，管理者との連絡が必要である．

表 3-7　授業削除機能の機能要件

機能要件
授業の削除はその授業の担当教員のみである
授業に問題の解答がある場合，削除の代わりに，非公開にする
担当教員は非公開した授業を閲覧できる

本システムの授業削除機能の実装では，実際のデータの削除が行われない．教員と授業を関連しているテーブルに Flag を追加する．削除の場合，Flag の値は True にする，非削除の場合は，Flag の値は False にする．Flag の値によって，授業を表示・非表示する．実際のデータの削除は管理者が行われる．

第4章　Web 剽窃チェック機能の開発

本章では，筆者は担当した Web 剽窃チェック機能の開発について紹介する．Web 剽窃チェックとは，学生の解答ソースコードはインターネットからコピーしたかどうかを調べることである．Web 剽窃チェックの結果，教員に関連ページを表示する．本システムで，関連ページとは，解答ソースコードのデータを含み，剽窃元と思われる Web ページである．誤検出を避けられないため, 学生の解答が Web から剽窃したかどうかの判断は最終的には教員が検出した関連ページにアクセスし確認することである．

4.1　Web 剽窃チェックの必要性

インターネットの普及で，インターネットから様々な情報手軽に入手できる．教育現場で使っているプログラミング課題は，インターネットでは既に大量な解答が公開している．教育において，盗用・剽窃は，学問のルールに反するだけでなく，場合によって他人の権利を侵害する犯罪行為である．一方，現場の教員にとっては，課題の採点するとき，人力でインターネットから盗用を検出するのは難しい．自動検出ツール・システムが必要である．

本プロジェクトでは，学生の剽窃行為を防止することを目的として，Web から関連ページを検出する機能を実装した．

4.2　剽窃パターン

プログラミングにおいて，どのような剽窃行為が行われるかいくつかまとめられている [14] [15]．筆者はこれらを参考にして，剽窃パターンをまとめた．

（1）完全コピー：本システムで提出するソースコードの出力データは特定の形式に合わせる必要がある．特定の形式とは，出力データの後には改行をつけないことである．本プロジェクトにおいては，完全コピーの定義は出力に合わせるため，出力変更以外，ほかのソースコードそのままコピーすることである．

（2）コメントの書き換え：Web からコピーしてきたソースコードのコメントを変更・追加し，ソースコードの見た目を変えることである．コメントの書き換えはプログラムの制御に影響を及ぼさない．

（3）インデントの変更：コメントの修正と同じように，ソースコードの見た目を変えることである．インデントの変更はプログラムの制御には影響を及ぼさない．

（4）順序に依存しない式の入れ替え： a++; b++; c++; のソースコードを c++; b++; a++ にするなど，プログラムの制御に影響を及ぼさない式の順序を入れ替えることである．

（5）定数名・変数名・関数名の変更：ソースコード中の定数，変数，関数の名前を変更するなど，プログラムの制御に影響を及ぼさないことである．

（6）式の書き換え：while 文を for 文にするといったプログラムの制御に影響を及ぼさない変更である．

（7）定数の変更：配列の長さ，メモリの大きさなどの場合で使われる定数の数字を変更することによって，プログラムの実行の結果が変わる可能だが，プログラムのロジック，制御に影響を及ぼさないことである．

（8）コードの分割：ソースコードの一部を A 関数としてまとめる，あるいは A 関数のソ

ースコードは別の B 関数に統合することである．

以上で紹介した剽窃パターンは 8 つがあったが，顧客にとっては，学生がどんな剽窃パターンでソースコードを修正したのかは，知る必要がない．一番大事なことは解答のソースコードは Web ページからコピーしていた場合，関連ページを検出することである．本プロジェクトでは，完全コピーの剽窃パターンは優先度が一番高く，対応すべき課題である．

4.3 技術・論文の調査と既存サービスの紹介

近年，各分野不正な剽窃チェックの研究が進んでいる．プログラムのソースコードの間での類似性を評価する手法がたくさん提案されている．使えるツールもたくさんある．Web ページからソースコードの盗用についての評価する手法が見つからない．Web ページから論文・レポートについての剽窃の検出手法がたくさん存在する[16][17][18].

論文・レポートに向けの Web サイトから剽窃チェックのオンラインサービスはたくさんある．一部を表 4-1 に示す

表 4-1 既存の剽窃チェックサービス

サービス名称	サービスリンク	制限
Duplichecker	http://www.duplichecker.com/	1000 文字以内
Plagiarism detect	http://plagiarism-detect.com/	100% free
Plagiarism detection	http://plagiarismdetection.org/	料金がかかる

筆者はこれらのサービスには，論文・レポートの代わり，Web サイトからコピーしたプログラミング問題の解答のソースコードを利用してテストした結果，関連ページを検出できなかった．いろいろなテストをした結果，これらのサービスはソースコードの剽窃チェックに対応できないと断定した．また，これらのサービスには手数料の制限，文字数の制限，外部にサービスの API を提供していないため，本研究プロジェクトには利用できないと断定した.

論文・レポートに向けの剽窃チェックサービス以外，筆者はソースコードの検索エンジンについて調査した. ソースコード向けの検索エンジンがたくさんある. 一部を表 4-2 に示す.

表 4-2 一部ソースコード検索エンジン

名称	サービスリンク	説明
Search code	https://searchcode.com/	Github,Bitbucket,GoogleCode,Codeplex,Sourceforge,Fedora Project などからソースコードを検索する
codota	http://www.codota.com	Android に限定している
Ohloh	http://code.openhub.net/	オープンソースのライセンスが正しく守られているかをチェックする

筆者は Web サイトからコピーしたプログラミング問題の解答のソースコードがこれらの

ソースコード検索エンジンでテストした結果，関連ページを検出できなかった．まだ提供している検索 API の使用回数の制限があり，日本語サポートをしていないため，本システムではこれらのサービスが利用できないと断定した．

4.4 Web 剽窃チェックの機能要件

Web 剽窃チェックの機能要件を表 4-3 に示す．本システムでは，剽窃チェックは授業の担当教員のみ利用できる．剽窃チェックの結果は解答ソースコードと関連あるページの情報を教員に提示するだけである．関連ページに同じソースコードを載っているかどうかについての分析は実装をしていない．剽窃しているかどうかの判断はシステムではなく，教員が関連ページにアクセスして確認することである．

表 4-3 Web 剽窃チェックの機能要件

機能要件
剽窃チェックは授業の担当教員のみ利用できる
評価が Accept の解答について剽窃チェックする
剽窃しているかどうかの判断はシステムではなく，教員が行う
剽窃チェックの利用はプログラミング問題の Web 剽窃チェック用のキーワードの作成が必要である
問題詳細ページと学生詳細ページでボタンを押すことで剽窃チェックを実行できる
剽窃チェック結果の表示画面にはユーザの解答情報と関連ページの情報を表示する
授業に参加する学生全員を一度に Web 剽窃チェックできる
授業に参加する学生１人ずつの Web 剽窃チェックできる
プログラミング問題の Web 剽窃チェック用のキーワードを変更したら，再チェックできる

剽窃チェックの利用は 2 つの前提がある．1 つ目は，プログラミング問題の Web 剽窃チェック用のキーワードの作成である．2 つ目は，解答の評価が Accept である．プログラミング問題の Web 剽窃チェック用のキーワードがない場合，関連ページの特定は難しくなる，精度もよくないため，プログラミング問題の Web 剽窃チェック用のキーワードの作成が必要である．一方，検索 API は使用料金がかかるため，全ての解答について剽窃チェックするのは検索 API 使用料金無駄になるため，Accept 以外の解答について，剽窃チェックが行われない．

4.5 提案手法

論文[17]によると，インターネットには膨大なデータが存在するので，インターネットからあるプログラミング問題の解答のソースコードを集め，手元のソースコードとの比較は現実ではない．一方，検索エンジンと手元の情報を利用して，剽窃元 Web ページを探すのは良いと述べた．本システムでは，学生が提出したソースコードと教員が作成した問題のデータを利用して検索用のキーワードを作成する，作成したキーワードで数回の API 検索をおこない，取得したデータから関連ページを探す．学生の解答は Web ページから完全コピーした剽窃パターンについて，筆者が考えた関連 Web ページを検出する手法を図 4-1 に示す．

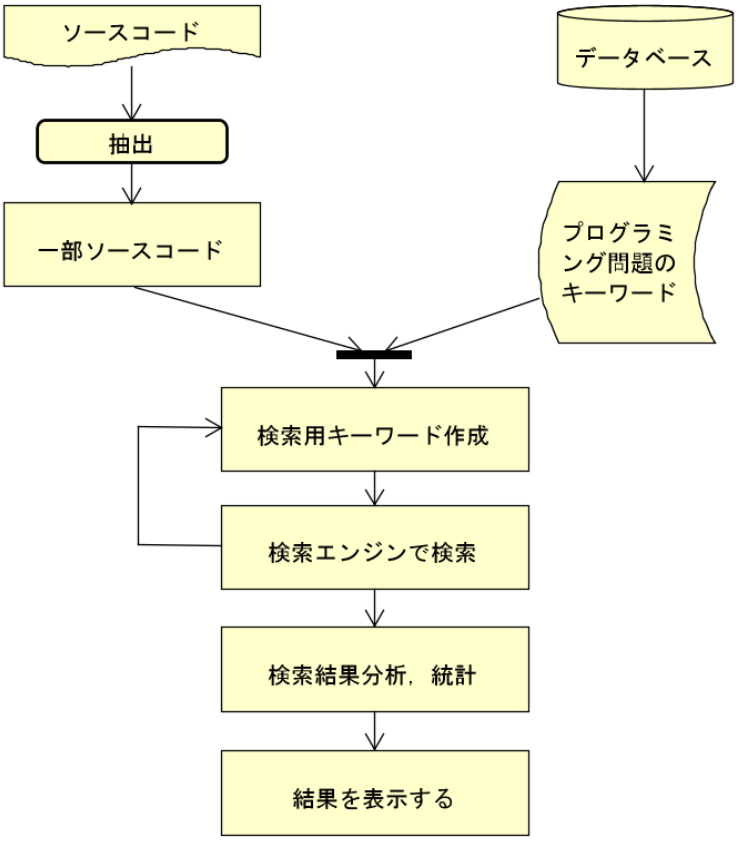

図 4-1 Web 剽窃チェックの流れ

これから完全コピーの剽窃パターンについて，Web 剽窃チェックの流れについて詳しく説明する．

（1）プログラミング問題の剽窃チェック用のキーワードを作成する．図 4-2 に示すように，教員はプログラミング問題新規作成するとき、インターネット剽窃チェック用のキーワードを作成する．プログラミング問題の剽窃チェック用のキーワードを作成しない場合，該当プログラミング問題では Web 剽窃チェックを利用できない．

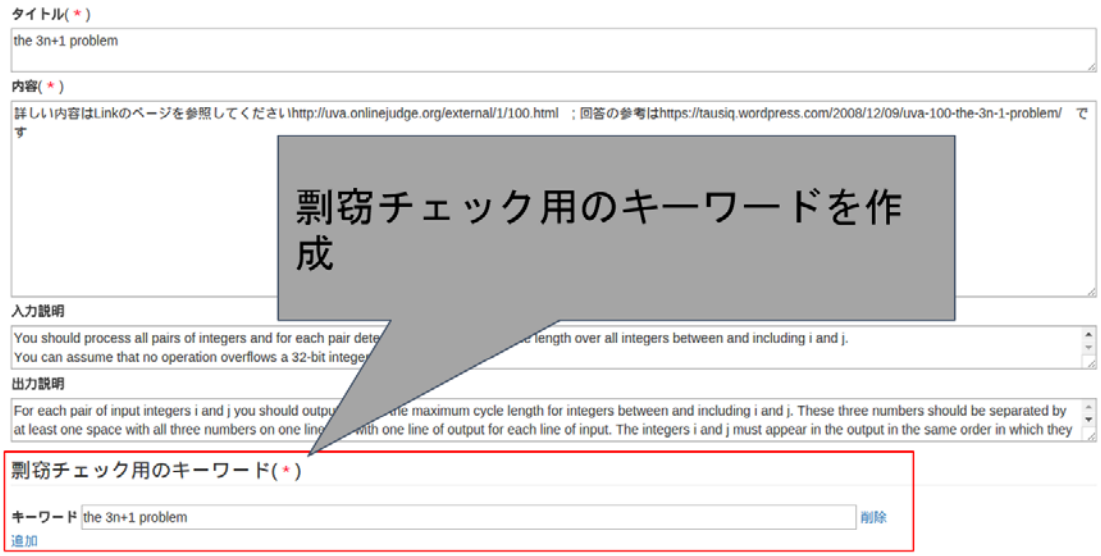

図 4-2　剽窃チェック用のキーワードの作成

（2）解答ソースコードから行単位で長い行のソースコードを抽出する．図 4-3 には行単位で長い行のソースコードの抽出を示している．ソースコード中から言語の汎用関数，言語処理系の行を削除し，特徴性があるソースコードを抽出する．C 言語で説明すると，「#include “stdio.h”」，「int main()」，「else」など利用しない．これらのソースコードで検索すると，多数の非関連ページが検出されるため，不採用にする．筆者が考えた特徴性があるソースコードとは長い行である．完全コピーの場合，長い行のソースコードは短い行のソースコードより，多い情報を含むと考えられる．当然，ソースコードのコメントも利用できる．しかし，コメントは自然言語で書いたので，各コメントの行が短い場合，作成したキーワードで検索すると，多数の非関連ページが検出され，関連ページの特定をできなくなる恐れがあるため，本提案手法では，長い行のソースコードを優先的に利用する．

```
#include <stdio.h>
int cycle_length(int n, int i)
{  int a;
    if (n == 1)
       return i;
    if(n%2 == 0)
       {a = cycle_length(n/2, ++i); return a;}
    else
       {a = cycle_length((3*n+1), ++i); return a;}
}

int main()
{
    int a, b, i, max, min;
    scanf("%d", &a);
    scanf("%d", &b);
    if (a >= b)
       {max = a; min = b;}
    else
        {max = b; min = a;}
    int c[max - min +1];

    for(i = 0; i < (max-min+1); i++)
    {
        c[i] = cycle_length((min+i), 1);
    }

    for (i=1; i <(max-min+1); i++)
    {
        if (c[i] >= c[0])
           {int t = c[0]; c[0] = c[i]; c[i] = t;}
}
printf("%d %d %d", a, b, c[0]);
return 0;
}
```

共通点がある行を削除(int main() else)
コメントを削除
長い行を利用する

```
a = cycle_length((3*n+1), ++i);
c[i] = cycle_length((min+i), 1);
for(i = 0; i < (max-min+1); i++)
int cycle_length(int n, int i);
a = cycle_length(n/2, ++i);
int c[max - min +1];
```

図 4-3　ソースコードの抽出

（3）プログラミング問題の剽窃チェック用のキーワードとソースコードを組み合わせて、検索用のキーワードを作成する．本提案手法は，1 回の剽窃チェックは 5 回の API 検索が行われるため，5 つの検索用のキーワードの作成が必要である．検索用のキーワードの作成ルールを表 4-4 に示す．

表 4-4　検索用のキーワードの作成ルール 1

検索用キーワード 1	Keyword SourcecodeA
検索用キーワード 2	Keyword SourcecodeA SourcecodeB
検索用キーワード 3	Keyword SourcecodeA SourcecodeB SourcecodeC
検索用キーワード 4	Keyword SourcecodeA SourcecodeB SourcecodeC SourcecodeD
検索用キーワード 5	Keyword SourcecodeA SourcecodeB SourcecodeC SourcecodeD SourcecodeE

Keyword はプログラミング問題の剽窃チェック用のキーワードである．SourcecodeA，SourcecodeB，SourcecodeC，SourcecodeD，SourcecodeE はプログラミング問題の解答ソースコードから抽出した 5 行のソースコードである．この中，SourcecodeA は一番長い行である． SourcecodeE は一番短い行である．そして，表 4-4 で示すやり方で作った検索用のキーワードを図 4-4 に示す．プログラミング問題の剽窃チェック用のキーワード「the 3n+1 problem」と解答のソースコードの一部を組み合わせて，5 つの検索用キーワードを作成した．

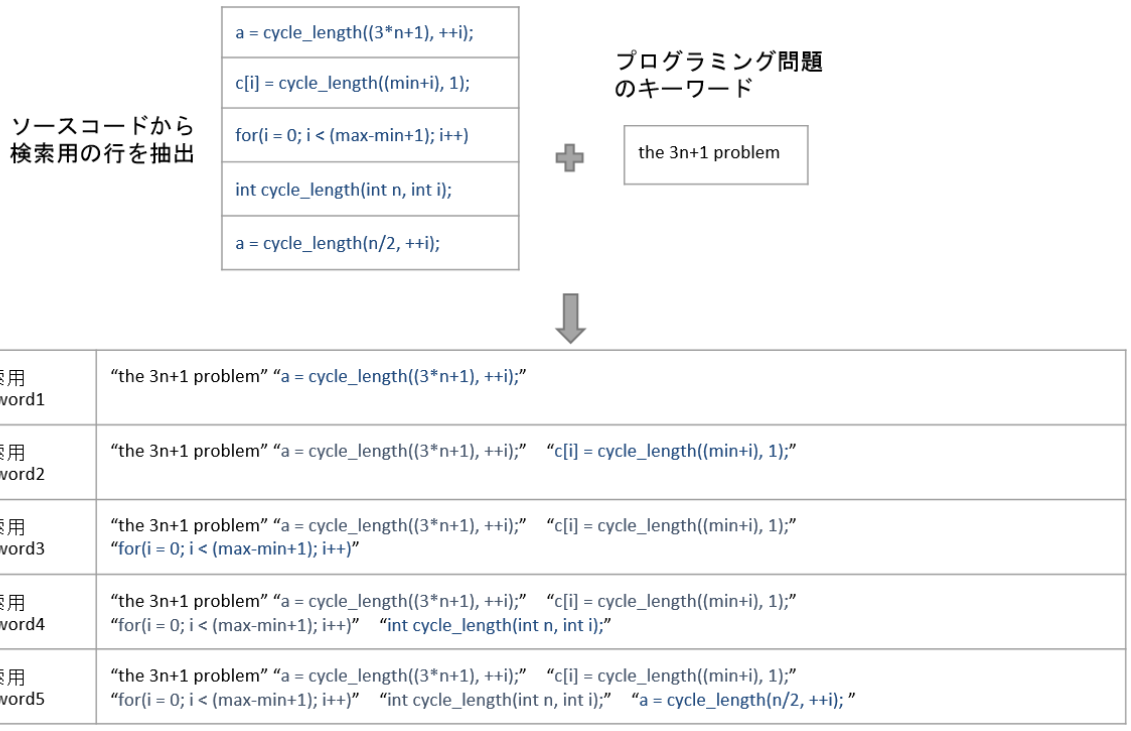

図 4-4　検索用キーワードの作成

（4）検索 API で検索用のキーワードで検索する．本システムでは，Microsoft の Bing 検索エンジンを利用している．Bing 検索エンジンは URL に Get アクセスをすると，XML や JSON で結果を返してくれる検索 API を提供している．作成した検索用キーワードを検索 API で検索を行い，取得したデータを図 4-5 に示す．検索 API で，5 回の検索で 10 件のデータを取得できる．各件のデータ中には Web サイトのタイトル，Web サイトのリンク，ID，Web 中身の説明などの情報を含む．

"https://api.datamarket.azure.com/Bing/Search/v1/Composite?Sources=%27Web%27&$format=json&Query=%27%22+the+3n%2B1+problem+%E8%A7%A3%E7%AD%94%22+%22while+%28+scanf+%28%22%25d+%25d%22%2C+%26i%2C+%26j%29+%22%3D+EOF+%29+%7B%22%27&Market=%27en-US%27&$top=10&$skip
=0"
{:__metadata=>
  {:uri=>
    "https://api.datamarket.azure.com/Data.ashx/Bing/Search/v1/Composite?Sources='Web'&Query='\" the 3n+1 problem 解答\" \"while (scanf (\"%25d %25d\", &i, &j) != EOF) {\"'&Market='en-US'&$skip=0&$top= ,
   :type=>"ExpandableSearchResult"},
 :ID=>"640f6596-24cb-43b2-ad06-908796b0bfb4",
 :WebTotal=>"969",
 :WebOffset=>"0",
 :ImageTotal=>"",
 :ImageOffset=>"",
 :VideoTotal=>"",
 :VideoOffset=>"",
 :NewsTotal=>"",
 :NewsOffset=>"",
 :SpellingSuggestionsTotal=>"",
 :AlteredQuery=>"",
 :AlterationOverrideQuery=>"",
 :Web=>
  [{:__metadata=>
     {:uri=>
       "https://api.datamarket.azure.com/Data.ashx/Bing/Search/v1/ExpandableSearchResultSet(guid'640f6596-24cb-43b2-ad06-908796b0bfb4')/Web?$skip=0&$top=1",
      :type=>"WebResult"},
    :ID=>"a4a00756-d1d7-4481-8837-40a9f557929b",
    :Title=>"c++ - what does scanf(\"%d %d\", &a, &b)==2 means - Stack ...",
    :Description=>
     "... My question is that in the below program,what does the code scanf(\"%d %d\", &a, &b)==2 do?? the program get two number and print the sum :) # ...",
    :DisplayUrl=>
     "stackoverflow.com/questions/25642013/what-does-scanfd-d-a-b-2-means",
    :Url=>
     "http://stackoverflow.com/questions/25642013/what-does-scanfd-d-a-b-2-means"},
   {:__metadata=>
     {:uri=>
       "https://api.datamarket.azure.com/Data.ashx/Bing/Search/v1/ExpandableSearchResultSet(guid'640f6596-24cb-43b2-ad06-908796b0bfb4')/Web?$skip=14&$top=1",
      :type=>"WebResult"},
    :ID=>"4fbb50a0-d02c-4fa1-a1e2-c953bb4fd206",
    :Title=>"c++ - How to further optimize this code for 3n + 1 problem ...",
    :Description=>
     "The problem can be viewed here, since it's styled. I don't think I should just copy and paste. And here's my code. I implemented recessive function and dynamic ...",
    :DisplayUrl=>
     "codereview.stackexchange.com/questions/20020/how-to-further...",
    :Url=>
     "http://codereview.stackexchange.com/questions/20020/how-to-further-optimize-this-code-for-3n-1-problem"},
   {:__metadata=>
     {:uri=>
       "https://api.datamarket.azure.com/Data.ashx/Bing/Search/v1/ExpandableSearchResultSet(guid'640f6596-24cb-43b2-ad06-908796b0bfb4')/Web?$skip=26&$top=1",
      :type=>"WebResult"},
    :ID=>"9a2cd1ba-e907-4e60-97bc-dac50c3c9bb9",
    :Title=>"UVa 100 (The 3n + 1 problem) | Solved Programming Problems",
    :Description=>
     "while (scanf(\"%d %d\", &i, &j) != EOF) {cout << i << \" \" << j << \" \" << algo(i, j) << endl;} return 0;} refencement google. December 23, 2012 at 2:45 am",
    :DisplayUrl=>
     "https://tausiq.wordpress.com/2008/12/09/uva-100-the-3n-1-problem",
    :Url=>
     "https://tausiq.wordpress.com/2008/12/09/uva-100-the-3n-1-problem/"},

図 4-5　検索 API で検索の結果

（5）数回の検索で集めたデータから関連ページを探す．検索結果の分析を図 4-6 に示す．本提案手法は，1 回の Web 剽窃チェックでは，全部 5 回の API 検索が行われる．毎回の API 検索の結果から Web サイトのタイトル，リンク，Web 中身の説明など情報を収集する．集計したデータから多く検出された Web サイトの情報を教員に表示する．繰り返し検出された Web サイトには解答ソースコードが入る可能性が高いと考えられる．

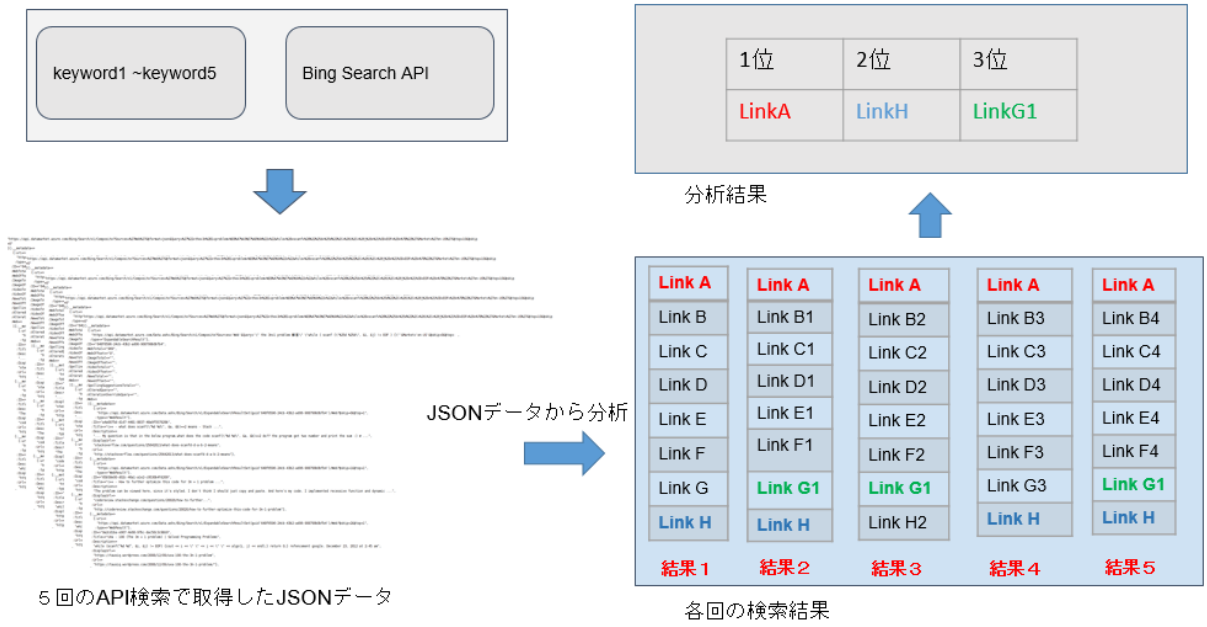

図 4-6　検索結果の分析

（6）剽窃元と関わる Web サイトの情報を表示する．図 4-7 に示すように，繰り返し検出された Web サイトには解答ソースコードが入る可能性が高いため，関連ページの情報を教員に表示する．剽窃したかどうかの判断は最終的には教員で判断する．また，剽窃チェックの結果を保存し，新しい解答が提出されるまで，あるいは，Web 剽窃チェック用のキーワードを変更するまで，保存されたデータを表示する．

図 4-7　剽窃チェックの結果の表示

4.6　Web 剽窃チェックの実装

4.6.1　Bing Search API の実装

本システムでは以下の手順で Bing Search API を実装した．

（1）APIKEY を取得する．Bing 検索 API を利用するまえ，Microsoft アカウントを作成し，APIKEY を取得する．APIKEY の取得アドレスは https://datamarket.azure.com/dataset/bing/search になる．

（2）ライブラリ searchbing を利用して実装する．ライブラリ searchbing には，Bing 検索 API の使用に関わる機能を実装した．利用者は，検索用のキーワード，検索の種類，取得データの件数を指定すれば，検索の結果を取得できる．

4.6.2　Ajax で剽窃チェックを実装

Web 剽窃チェックの機能要件に「授業に参加する学生 1 人ずつの Web 剽窃チェックできる」と言う要件がある．図 4-8 は学生詳細ページを示す．評価が Accept の解答について，Web 剽窃チェックの利用は可能になる．本システムでは，1 回の剽窃チェックは検索 API を利用し，5 回の API 検索が行われる．毎回の API 検索は時間がかかるため，Ajax を使ってデータの通信をバックエンド側にやることで，WEB フロント側には別の動作が行われる．バックエンド側のデータ解析を終わったら，結果を WEB フロント側に反映する．Web 剽窃チェックでは，Ajax を利用し，Web アプリケーションのパフォーマンスを改善した．

図 4-8　1 つずつの剽窃チェック

4.6.3　Active job の利用

Web 剽窃チェックの機能要件に「授業に参加する学生全員を一度に Web 剽窃チェックできる」と言う項目がある．図 4-9 は問題詳細ページの全員解答状況を示す．このページで，授業に参加する学生全員を一度に Web 剽窃チェックが行われる．授業に参加する学生が多い場合，全員の剽窃チェックするのは時間がかかる．また，本システムでは，解答ソースコードの評価，ローカル剽窃チェックなども行われる．これらの動作が同時に行われる場合，サーバの動作が重くなる，システムのパフォーマンスが落ちる．この可能性を下げるため，

Active job を導入する．Active job は，ジョブを宣言し，それぞれによってバックエンドで様々な方法によるキュー操作を実行するためのフレームワークです．これらのジョブでは，優先度がある．多数の動作が同時に行われるとき，優先度が高いジョブは優先実行する．優先度が低いジョブは後で実行する．ジョブの優先度によって，システムのパフォーマンスを改善する．

図 4-9 問題詳細ページの全員解答状況

4.7 工夫したところ

4.7.1 検索用キーワードの作成と検索回数を減らす

筆者が最初考えた検索用のキーワードの作成ルールを表 4-5 に示す．Keyword はプログラミング問題の剽窃チェック用のキーワードである．SourcecodeA，SourcecodeB，SourcecodeC，SourcecodeD，SourcecodeE はプログラミング問題の解答ソースコードから抽出した 5 行のソースコードである．表 4-5 の一行目を例として説明すると，Keyword と SourcecodeA を合わせて，1 つの検索用のキーワード 1 になる．

表 4-5 検索用キーワードの作成ルール 2

検索用のキーワード 1	Keyword SourcecodeA
検索用のキーワード 2	Keyword SourcecodeB
検索用のキーワード 3	Keyword SourcecodeC
検索用のキーワード 4	Keyword SourcecodeD
検索用のキーワード 5	Keyword SourcecodeE

しかし，表 4-5 に示すやり方で以下の問題を存在する．検索用のキーワードと検索 API を利用して検索するとき，検索エンジンから検索結果を返す．1 回の API 検索は前 10 件のデータだけ取得する．次の 10 件データを取るとき，もう 1 回の API 検索が必要になる．検索

エンジンのアルゴリズムによって，関連ページは検索結果の前何件にいるのは確認できないため,最初から何件までデータを収集するのをきめられない.検索結果の精度が上がるため,多くのデータの収集が必要である．しかし，検索 API の使用回数を増えると，たくさんの使用料金を発生する．そして，表 4-5 に示すやり方より，もっと効率的な検索用のキーワードの作成が必要になる．

表 4-4 に示すやり方で，1 回目以降の検索は前回で利用したキーワードの上に，1 行新しいソースコードを追加し，検索用のキーワードを作成する．検索は AND の演算を利用しているため，1 回目以降の検索結果は 1 回目の検索結果より精度が高く，非関連ページのデータが少なくなる．関連ページは検索結果の上位に出る可能性が高くなる．テストの結果，検索回数が減る，検索精度が高くなる傾向があった．

4.7.2 検索エンジンの検索テクニックの活用

検索エンジンの検索テクニックはたくさんがある．本システムで活用した検索テクニックは 2 つがある．

1 つ目は検索エンジンの演算子 AND を利用する．AND 演算は複数のキーワードを含むページを検索できる．普段検索する時，AND 演算子はよく使われている．キーワードとキーワードの間にスペースを入れる，または AND 入れることで AND 演算になる．スペースと AND での検索結果はあまり変わらない．本システムで，表 4-4 に示すやり方で検索用のキーワードを作成する．プログラミング問題の剽窃チェック用のキーワードとソースコードの間は AND 演算でつながる．AND 演算で検索すると，検索回数が減り，関連ページは上位に上がる効果があった．

2 つ目はダブルクォーテーション（""）を利用する．ダブルクォーテーション（""）でキーワードを囲う場合，完全一致検索を行う．ダブルクォーテーション内のキーワードを順番どおりに含むページだけが表示される．Microsoft の Bing 検索エンジンはダブルクォーテーション（""）での検索は随分優れている．ダブルクォーテーションを使うかどうか，検索の結果は随分変わる．本システムでは，ダブルクォーテーションを利用し，検索の精度がよくなった．

第5章　評価実験

本章では，提案手法の有効性と提案手法が対応できる剽窃パターンについて検証する．剽窃パターンに対応できるかできないかの判断基準は関連ページを検出できるか否かのことである．

5.1　目的

本実験の目的は 2 つがある．

（1）提案手法の有効性について検証する．

（2）対応できる剽窃パターンについて検証する．

提案手法の有効性を検証するため，以下 4 つの項目について，実験を行った．

項目 1：完全コピーの場合，Web 剽窃チェックの成功率．

項目 2：完全コピーの場合，検索用のキーワードの変化による関連ページの検出精度の変化．

筆者が提案した Web 剽窃チェック手法では，1 回の剽窃チェック，5 つの検索用のキーワードを作成し，5 回の API 検索が行われる．それぞれの検索用のキーワードで，関連ページの検出率を求める．検索用のキーワードの長さと検出率の関係を検証する．

項目 3：完全コピーの場合，日付が変わるによる，剽窃チェックの成功率への影響．

項目 4：完全コピーの場合，剽窃チェック用のキーワードの変更による，剽窃チェックの成功率の変化．

提案手法で，対応できる剽窃パターンを検証するため，以下 1 つの項目について実験を行った．

項目 5：剽窃パターンによる変更した実験データで関連ページの検出率を集計する．

5.2　利用するデータ

今回の実験で利用しているデータを表 5-1 に示す．プログラミング問題は 4 問がある．「the 3n+1 problem」問題は 10 件の解答を集めた．「jolly jumper」問題は 9 件の解答を集めた．「Financial Management」問題と「GCD and LCM」問題は 1 件の解答を集めた．これらのプログラミング問題は全部プログラミングコンテストサイトの問題である．表 5-1 に示す解答は全て Web サイトからコピーしてきたものである．これらの解答のソースコードは出力変更以外，ほかのソースコードそのままコピーしたことである．筆者はこれらのプログラミング問題は本システムで新規作成した．これらの解答は本システムで評価し，Accept になった解答である．

表 5-1　実験データ一覧

プログラミング問題	解答	言語
the 3n+1 problem	solution1	C
	solution2	C
	solution3	C++
	solution4	C++
	solution5	C++
	solution6	C++
	solution7	C++
	solution8	C++
	solution9	C++
	solution10	C++
jolly jumper	solution11	C++
	solution12	C++
	solution13	C++
	solution14	C++
	solution15	C++
	solution16	C++
	solution17	C++
	solution18	C
	solution19	C
Financial Management	solution20	C
GCD and LCM	solution21	Python

5.3　評価方法

一般的に良く使われる評価指標に, Precision, Recall, Specificity, F-measure, Accuracy, AUC がある．今回の実験には，Precision[19]の評価指標を利用した．Precision（精度）は，検索結果の中にどの程度正解が含まれるかを示す. 今回の実験の評価モデルを表 5-2 に示す. 関連ページは解答ソースコードと同じコードを載っているページである．非関連ページは解答ソースコードと違うコードを載っているページである.

表 5-2　評価モデル

	関連ページ	非関連ページ
検出される	TP	FP
検出されない	FN	TN

表 5-2 に示す評価モデルの基に，Precision は以下の式で表示する

$$\text{Precision} = \frac{TP}{TP + FP}$$

5.4　　実験データの収集

今回の実験には，2 種類のデータを収集した．

第 1 種類のデータは Web 剽窃チェックの結果データである．Web 剽窃チェックの結果のデータは 5 回の API 検索で取得したデータのまとめである．サンプルデータを図 5-1 に示す．図中の title は関連ページのタイトルである．Link は関連ページのリンクである．Times は 5 回の API 検索で，何回検出されることを示す．

title	link	times
UVa : 100 (The 3n + 1 problem) | Solved Programming Problems	https://tausiq.wordpress.com/2008/12/09/uva-100-the-3n-1-problem/	5
Programming Challenge: The 3n+1 problem | LoBlog	https://loblog.wordpress.com/2007/03/29/programming-challenge-the-3n1-problem/	3
c - UVA's 3n+1 wrong answer although the test cases are ...	http://stackoverflow.com/questions/9624931/uvas-3n1-wrong-answer-although-the-test-cases-are-correct	3
核心网络—杭电OJ——1032 The 3n + 1 problem	http://www.netfoucs.com/article/lishuhuakai/23879.html	3
c++ - How to further optimize this code for 3n + 1 problem ...	http://codereview.stackexchange.com/questions/20020/how-to-further-optimize-this-code-for-3n-1-problem	2
C语言，The 3n + 1 problem 看我写的代码 ... - 知道	http://zhidao.baidu.com/question/343270143.html	2
c++ - Help with a programming challenge - Stack Overflow	http://stackoverflow.com/questions/6629802/help-with-a-programming-challenge	2
Caramba, no site http://acm.uva.es/problemset os caras ...	http://www.hardware.com.br/comunidade/caramba-site/167441/	2
hdoj 1032 The 3n + 1 problem - CSDN博客	http://m.blog.csdn.net/blog/flyupliu/16905297	1
The 3n + 1 Problem In C++ - C And C++ | Dream.In.Code	http://www.dreamincode.net/forums/topic/248963-the-3n-1-problem-in-c/	1
Uva 100 – The 3n + 1 | Shipu's Blog	http://shipuahamed.com/2012/08/uva-100-the-3n-1.html	1
hdoj1032 3n+1问题 —核心网络 - demo.netfoucs.com	http://demo.netfoucs.com/shengweisong/article/details/27070839	1
沒有標題阿~~標題很重要嗎	http://danielin02030.blogspot.com/	1
Nothing is everything: 【Q100】3n+1 Problem	http://new-acos.blogspot.com/2007/02/q1003n1-problem.html	1
The 3n + 1 problem - CSDN博客	http://m.blog.csdn.net/blog/evi__/18982741	1
杭电OJ——1032 The 3n + 1 problem - xinyuyuanm - 博客园	http://www.cnblogs.com/xinyuyuanm/p/3177696.html	1
1410131848-hd-The 3n + 1 problem - 网络二师兄 - CSDN.NET	http://blog.csdn.net/wangluoershixiong/article/details/40049963	1
hdu 1032 The 3n + 1 problem - 风华绝代 - 博客频道 - CSDN.NET	http://blog.csdn.net/were__wolf/article/details/10593561	1
The 3n + 1 problem - Arcfat Tsui - 博客园	http://www.cnblogs.com/arcfat/archive/2012/11/10/2764508.html	1

図 5-1　Web 剽窃チェックの集計結果

第 2 種類のデータは毎回の検索 API で取得したデータの記録である．サンプルデータを図 5-2 に示す．図中の keyword は検索用のキーワードである．title は関連ページのタイトルである．Link は関連ページのリンクである．Times は全部 1 である．

keyword	title	link	times
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"	c++ - How to further optimize this code for	http://codereview.stackexchange.com/questions/20020/how-to-further-o	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"	UVa : 100 (The 3n + 1 problem) | Solved Pr	https://tausiq.wordpress.com/2008/12/09/uva-100-the-3n-1-problem/	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"	Programming Challenge: The 3n+1 problem	https://loblog.wordpress.com/2007/03/29/programming-challenge-the-3n	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"	c - UVA's 3n+1 wrong answer although the	http://stackoverflow.com/questions/9624931/uvas-3n1-wrong-answer-al	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"	hdoj 1032 The 3n + 1 problem - CSDN博客	http://m.blog.csdn.net/blog/flyupliu/16905297	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"	C语言，The 3n + 1 problem 看我写的代码 .	http://zhidao.baidu.com/question/343270143.html	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"	The 3n + 1 Problem In C++ - C And C++ | D	http://www.dreamincode.net/forums/topic/248963-the-3n-1-problem-in-c/	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"	核心网络—杭电OJ——1032 The 3n + 1 prob	http://www.netfoucs.com/article/lishuhuakai/23879.html	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"	Uva 100 – The 3n + 1 | Shipu's Blog	http://shipuahamed.com/2012/08/uva-100-the-3n-1.html	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"	c++ - Help with a programming challenge -	http://stackoverflow.com/questions/6629802/help-with-a-programming-c	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)"	c++ - How to further optimize this code for	http://codereview.stackexchange.com/questions/20020/how-to-further-o	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)"	Programming Challenge: The 3n+1 problem	https://loblog.wordpress.com/2007/03/29/programming-challenge-the-3n	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)"	100 (The 3n + 1 problem) - Solved Program	https://tausiq.wordpress.com/2008/12/09/uva-100-the-3n-1-problem/	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)"	c - UVA's 3n+1 wrong answer although the	http://stackoverflow.com/questions/9624931/uvas-3n1-wrong-answer-al	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)"	核心网络—杭电OJ——1032 The 3n + 1 prob	http://www.netfoucs.com/article/lishuhuakai/23879.html	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)"	c++ - Help with a programming challenge -	http://stackoverflow.com/questions/6629802/help-with-a-programming-c	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)"	hdoj1032 3n+1问题 —核心网络 - demo.netfo	http://demo.netfoucs.com/shengweisong/article/details/27070839	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)"	沒有標題阿~~標題很重要嗎	http://danielin02030.blogspot.com/	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)"	Nothing is everything: 【Q100】3n+1 Proble	http://new-acos.blogspot.com/2007/02/q1003n1-problem.html	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)"	Caramba, no site http://acm.uva.es/problem	http://www.hardware.com.br/comunidade/caramba-site/167441/	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)" "if (n % 2 == 1) n = 3 * n + 1;"	Programming Challenge: The 3n+1 problem	https://loblog.wordpress.com/2007/03/29/programming-challenge-the-3n	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)" "if (n % 2 == 1) n = 3 * n + 1;"	100 (The 3n + 1 problem) - Solved Program	https://tausiq.wordpress.com/2008/12/09/uva-100-the-3n-1-problem/	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)" "if (n % 2 == 1) n = 3 * n + 1;"	The 3n + 1 problem - CSDN博客	http://m.blog.csdn.net/blog/evi__/18982741	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)" "if (n % 2 == 1) n = 3 * n + 1;"	c - UVA's 3n+1 wrong answer although the	http://stackoverflow.com/questions/9624931/uvas-3n1-wrong-answer-al	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)" "if (n % 2 == 1) n = 3 * n + 1;"	核心网络—杭电OJ——1032 The 3n + 1 prob	http://www.netfoucs.com/article/lishuhuakai/23879.html	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)" "if (n % 2 == 1) n = 3 * n + 1;"	杭电OJ——1032 The 3n + 1 problem - xinyu	http://www.cnblogs.com/xinyuyuanm/p/3177696.html	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)" "if (n % 2 == 1) n = 3 * n + 1;"	1410131848-hd-The 3n + 1 problem - 网络二	http://blog.csdn.net/wangluoershixiong/article/details/40049963	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)" "if (n % 2 == 1) n = 3 * n + 1;"	C语言，The 3n + 1 problem 看我写的代码 .	http://zhidao.baidu.com/question/343270143.html	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)" "if (n % 2 == 1) n = 3 * n + 1;"	hdu 1032 The 3n + 1 problem - 风华绝代 - 博	http://blog.csdn.net/were__wolf/article/details/10593561	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)" "if (n % 2 == 1) n = 3 * n + 1;"	The 3n + 1 problem - Arcfat Tsui - 博客园	http://www.cnblogs.com/arcfat/archive/2012/11/10/2764508.html	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)" "if (n % 2 == 1) n = 3 * n + 1;" "max_cycle_length = cycle_length"	UVa : 100 (The 3n + 1 problem) | Solved Pr	https://tausiq.wordpress.com/2008/12/09/uva-100-the-3n-1-problem/	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)" "if (n % 2 == 1) n = 3 * n + 1;" "max_cycle_length = cycle_length"	Caramba, no site http://acm.uva.es/problem	http://www.hardware.com.br/comunidade/caramba-site/167441/	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {" "if (cycle_length > max_cycle_length)" "if (n % 2 == 1) n = 3 * n + 1;" "max_cycle_length = cycle_length"	UVa : 100 (The 3n + 1 problem) | Solved Pr	https://tausiq.wordpress.com/2008/12/09/uva-100-the-3n-1-problem/	1

図 5-2　検索 API で返したデータのまとめ

5.5　　結果

5.5.1 項目 1 について

項目 1 は， Web 剽窃チェックの成功率を求める．実験で第 1 種類のデータを収集した．各解答の Web 剽窃チェック結果の集計を表 5-3 に示す．剽窃チェックの結果は 0，1 で表示する．値 1 は，関連ページが検出されたことを示す．値 0 は，関連ページが検出されなかったことを示す．

表 5-3 を示すように，「the 3n+1 problem」の問題は 10 件の解答があって，検出できた件数は 8 件，成功率は 80%である．「jolly jumper」の問題は 9 件の解答があって，検出できた件数は 5 件, 成功率は 55.6%である. 全部 21 件の解答があった. 成功は 15 件, 失敗は 6 件. 成功率は 71.4%である．

表 5-3 各解答の Web 剽窃チェック結の果集計

プログラミング問題	解答	言語	剽窃チェックの結果
the 3n+1 problem	solution1	C	1
	solution2	C	1
	solution3	C++	1
	solution4	C++	1
	solution5	C++	1
	solution6	C++	1
	solution7	C++	0
	solution8	C++	0
	solution9	C++	1
	solution10	C++	1
jolly jumper	solution11	C++	1
	solution12	C++	1
	solution13	C++	1
	solution14	C++	0
	solution15	C++	1
	solution16	C++	1
	solution17	C++	0
	solution18	C	0
	solution19	C	0
Financial Management	solution20	C	1
GCD and LCM	solution21	Python	1

5.5.2 項目 2 について

項目 2 には，検索用のキーワードの長さと関連ページの検出率の関係について検証する．実験で，第 2 種類のデータを収集した．5.3 に紹介した評価方法で検索用のキーワードの長さと関連ページの検出率の関係について検証した．

まず，「the 3n+1 problem」問題と solution1 の解答で例を挙げて説明する．図 5-3 は，1 回目の API 検索で取得したデータからまとめたデータを示す. 全文 10 件のデータがあった. 各 Web サイトにアクセスして確認したところ, 黄色い背景の Link は関連ページと判明した.

関連ページは 2 件がある．非黄色い背景の Link は非関連ページである

keyword	title	link	times
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"	100 – The 3n + 1 problem Solutio	https://fabhodev.wordpress.com/2015/03/18/100-the-3n-1-problem-solution/	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"	UVa : 100 (The 3n + 1 problem)	**https://tausiq.wordpress.com/2008/12/09/uva-100-the-3n-1-problem/**	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"	c - UVA's 3n+1 wrong answer alth	http://stackoverflow.com/questions/9624931/uvas-3n1-wrong-answer-although-the-test-cases-are-corr	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"	Programming Challenge: The 3n+	**https://loblog.wordpress.com/2007/03/29/programming-challenge-the-3n1-problem/**	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"	c++ - How to further optimize this	http://codereview.stackexchange.com/questions/20020/how-to-further-optimize-this-code-for-3n-1-prob	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"	hdoj 1032 The 3n + 1 problem - C	http://m.blog.csdn.net/blog/flyupliu/16905297	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"	C语言，The 3n + 1 problem 看我	http://zhidao.baidu.com/question/343270143.html	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"	The 3n + 1 Problem In C++ - C A	http://www.dreamincode.net/forums/topic/248963-the-3n-1-problem-in-c/	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"	Uva 100 – The 3n + 1	Shipu's B	http://shipuahamed.com/2012/08/uva-100-the-3n-1.html	1
" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"	c++ - Help with a programming c	http://stackoverflow.com/questions/6629802/help-with-a-programming-challenge	1

図 5-3　1 回目の API 検索で取得したデータ

5.3 に紹介した評価方法，モデルを利用すると．1 回の API 検索で 10 件のデータを取得した．検索用のキーワード「" the 3n+1 problem" "while (scanf ("%d %d", &i, &j) != EOF) {"」を利用し，API 検索で取得したデータを分析した，表 5-4 を示すようの結果があった．この検索用のキーワードで関連ページの検出精度 Precision は 20%である．

表 5-4　1 回目の API 検索で取得したデータの分析結果

″ the 3n+1 problem″ ″while (scanf (″%d %d″, &i, &j) != EOF) {″		
	関連ページ	非関連ページ
検出する	2	8
検出しない	0	0

次，「the 3n+1 problem」問題の solution1 について，5 回の API 検索で取得したデータから各回の関連ページの検出精度 Precision を計算した．集計の結果を表 5-5 に示す．keyword A などキーワードについての説明は第 4 章の表 4-4 で紹介した．

表 5-5　毎回 API 検索の検出精度

	keywordA	keywordAB	keywordABC	keywordABCD	keywordABCDE
solution1	20	20	20	50	100

最後，全ての実験データについて 5 回の実験を行った．各検索用のキーワードでの平均検出精度 Precision のまとめを表に示す．値 0 の場合，API 検索で取得したデータがあるが，Web ページにアクセスして確認したところ，解答のソースコードと同じコードを載っていない．値が(-)の場合，API 検索で取得したデータがなかった．

全ての解答の各回の API 検索の検出精度のまとめを表 5-6 に示す．値 100 は，検索用のキーワードで取得したデータには，関連ページだけ検出されたことを表す．集計の結果からみると，検索用のキーワードが長くなると，関連ページの検出精度が上がる傾向がある．一方，関連ページの検出精度が下がった場合も存在する．

表 5-6　全ての解答の各回の API 検索の検出精度

プログラミング問題	解答	言語	各回の検索用のキーワード				
			Keyword A	Keyword AB	Keyword ABC	Keyword ABCD	Keyword ABCDE
the 3n+1 problem	solution1	C	20	20	20	50	100
	solution2	C	100	100	18.7	20	41.9
	solution3	C++	100	100	10	10	14.3
	solution4	C++	50	100	100	100	100
	solution5	C++	28.6	50	100	100	100
	solution6	C++	100	100	100	100	100
	solution7	C++	0	0	(–)	(–)	(–)
	solution8	C++	(–)	(–)	(–)	(–)	(–)
	solution9	C++	66.7	100	100	100	100
	solution10	C++	16.7	100	100	100	100
jolly jumper	solution11	C++	100	10	100	100	100
	solution12	C++	100	100	100	100	100
	solution13	C++	100	100	100	100	100
	solution14	C++	(–)	(–)	(–)	(–)	(–)
	solution15	C++	100	100	100	100	100
	solution16	C++	100	100	100	100	100
	solution17	C++	0	10	50	(–)	(–)
	solution18	C	0	0	0	0	(–)
	solution19	C	(–)	(–)	(–)	(–)	(–)
Financial Management	solution20	C	100	100	100	100	100
GCD and LCM	solution21	Python	20	100	100	100	100

5.5.3　　項目 3 について

項目 3 には，日付が変わるによる，剽窃チェックの成功率への影響について検証する．項目 1 の実験を連続 5 日で行った．集計した結果を表 5-7 に示す．剽窃チェックの結果は 0，1 で表示する．値 1 は，剽窃チェックの結果，関連ページが検出されたことを表す．値 0 は，剽窃チェックの結果,関連ページが検出されなかったことを表す.集計した結果からみると，同じプログラミング問題の解答について，短時間内，関連ページの検出率が変わらないという結果があった.

表 5-7　連続 5 日の Web 剽窃チェックの結果集計

プログラミング問題	解答	言語	1 日目剽窃チェックの結果	2 日目剽窃チェックの結果	3 日目剽窃チェックの結果	4 日目剽窃チェックの結果	5 日目剽窃チェックの結果
the 3n+1 problem	solution1	C	1	1	1	1	1
	solution2	C	1	1	1	1	1
	solution3	C++	1	1	1	1	1
	solution4	C++	1	1	1	1	1
	solution5	C++	1	1	1	1	1
	solution6	C++	1	1	1	1	1
	solution7	C++	0	0	0	0	0
	solution8	C++	0	0	0	0	0
	solution9	C++	1	1	1	1	1
	solution10	C++	1	1	1	1	1
jolly jumper	solution11	C++	1	1	1	1	1
	solution12	C++	1	1	1	1	1
	solution13	C++	1	1	1	1	1
	solution14	C++	0	0	0	0	0
	solution15	C++	1	1	1	1	1
	solution16	C++	1	1	1	1	1
	solution17	C++	0	0	0	0	0
	solution18	C	0	0	0	0	0
	solution19	C	0	0	0	0	0
Financial Management	solution20	C	1	1	1	1	1
GCD and LCM	solution21	Python	1	1	1	1	1

5.5.4　　項目 4 について

項目 4 には，プログラミング問題の Web 剽窃チェック用のキーワードの変更による，剽窃チェックの成功率への影響について検証する．筆者はプログラミング問題ごとで実験を行った．

「the 3n+1 problem」の問題の集計結果を表 5-8 に示す．値 1 は，剽窃チェックの結果，関連ページが検出されたことを表す．値 0 は，剽窃チェックの結果，関連ページが検出されなかったことを表す．( ) 中の値は，5 回の API 検索で，関連ページが何回検出されたことを表す．値「5/5」は，5 回の API 検索で，関連ページが 5 回検出されたことを表す．( ) がない場合，関連ページが検出されないことを表す．

表 5-8　the 3n+1 problem の集計結果

プログラミング問題	解答	言語	Web 剽窃チェック用のキーワード			
			the 3n+1 problem	the 3n+1 problem 解答	the 3n+1 problems	なし
the 3n+1 problem	solution1	C	1(5/5)	1(5/5)	1(5/5)	1(4/5)
	solution2	C	1(5/5)	0	0	1(5/5)
	solution3	C++	1(5/5)	0	0	1(5/5)
	solution4	C++	1(5/5)	0	0	1(5/5)
	solution5	C++	1(5/5)	0	0	1(4/5)
	solution6	C++	1(5/5)	0	0	1(5/5)
	solution7	C++	0(1/5)	0	0	1(5/5)
	solution8	C++	0	0	0	0
	solution9	C++	1(5/5)	0	0	1(5/5)
	solution10	C++	1(5/5)	0	0	1(4/5)

「jolly jumper」の問題の集計結果を表 5-9 に示す．値の意味は表 5-8 と同じである．

表 5-9　jolly jumper の集計結果

プログラミング問題	解答	言語	Web 剽窃チェック用のキーワード			
			jolly jumper	jolly jumper 解答	jolly jumper solution	なし
jolly jumper	solution11	C++	1(5/5)	1(3/5)	1(3/5)	1(4/5)
	solution12	C++	1(5/5)	0	0	1(4/5)
	solution13	C++	1(5/5)	0	0	1(5/5)
	solution14	C++	0(0/5)	0	0	0(0/5)
	solution15	C++	1(5/5)	0	0	0(0/5)
	solution16	C++	1(5/5)	1(5/5)	1(5/5)	1(5/5)
	solution17	C++	0(1/5)	1(3/5)	1(3/5)	1(5/5)
	solution18	C	0	0	0	0
	solution19	C	0	0	0	0

「Financial Management」の問題の集計結果を表 5-9 に示す．値の意味は表 5-8 と同じである．

表 5-10　Financial Management の集計結果

プログラミング問題	解答	言語	Web 剽窃チェック用のキーワード			
			Financial Management	Financial Management C 言語	the 3n+1 problem	なし
Financial Management	solution20	C	1(5/5)	0	0	1(3/5)

「GCD and LCM」の問題の集計結果を表 5-9 に示す．値の意味は表 5-8 と同じである．

表 5-11　GCD and LCM の集計結果

プログラミング問題	解答	言語	Web 剽窃チェック用のキーワード			
			最大公約数　最小公倍数	GCD LCM	the 3n+1 problem	なし
GCD and LCM	solution21	Python	1(5/5)	1(4/5)	1(4/5)	1(3/5)

集計結果から見ると， Web 剽窃チェック用のキーワードは剽窃チェックの結果に影響があった．Web 剽窃チェック用のキーワードを利用しなくても，関連ページが検出された．表 5-8 と表 5-9 の色付けのセルを見ると，Web 剽窃チェック用のキーワードを利用しない方は利用する方より，多くの関連ページが検出された．しかし，5 回の API 検索で，関連ページの検出率が減っている．本システムでは，プログラミング問題の Web 剽窃チェック用のキーワードがない場合，Web 剽窃チェック機能を利用できない．今後，この問題について検討する必要がある．

5.5.5　　項目 5 について

項目 5 には，剽窃パターンによる変更した実験データで，関連ページが検出されるかどうかについて検証する．

本提案手法は，ソースコードのコメントを利用しない，キーワードを作成するまえ，インデントを削除するなど処理があったため，剽窃パターンのコメントの書き換え，インデントの変更は検索用のキーワードの作成には影響がない．剽窃結果は完全コピーと同じ結果である．順序に依存しない式の入れ替え，定数名・変数名・関数名の変更，定数の変更，式の書き換えの剽窃パターンについての検証が必要である．

利用したデータは，収集した実験データが剽窃パターンの基に加工し，作成したデータである． 実験の結果の集計を表 5-12 に示す．値 1 は，剽窃チェックの結果，関連ページが検出されたことを表す．値 0 は，関連ページが検出されなかったことを表す．定数の変更の列には，(-) の値があった．(-) の値は該当解答には定数の定義していないため，実験を行わなかったことを示す．

表 5-12　剽窃パターンについての検証の結果

プログラミング問題	解答	言語	Web 剽窃チェックの結果				
			完全コピー	順序に依存しない式の入れ替え	定数名・変数名・関数名の変更	定数の変更	式の書き換え
the 3n+1 problem	solution1	C	1	1	0	(−)	0
	solution2	C	1	1	0	(−)	0
	solution3	C++	1	1	0	(−)	0
	solution4	C++	1	1	0	0	0
	solution5	C++	1	1	0	(−)	0
	solution6	C++	1	1	0	1	0
	solution7	C++	0	0	0	(−)	0
	solution8	C++	0	0	0	0	0
	solution9	C++	1	1	0	(−)	0
	solution10	C++	1	1	0	(−)	0
jolly jumper	solution11	C++	1	1	0	0	0
	solution12	C++	1	1	0	(−)	0
	solution13	C++	1	1	0	0	0
	solution14	C++	0	0	0	0	0
	solution15	C++	1	1	0	0	0
	solution16	C++	1	1	0	0	0
	solution17	C++	0	0	0	0	0
	solution18	C	0	0	0	0	0
	solution19	C	0	0	0	0	0
Financial Management	solution20	C	1	1	0	(−)	0
GCD and LCM	solution21	Python	1	1	0	(−)	0

表 5-12 の色付けのセルをみると，定数の変更の剽窃チェックには，関連ページが検出された場合もある．作成した検索用のキーワードを確認すると，定数と関わるコードが利用されていないことは明らかになった．集計結果から見ると，本提案手法は，定数名・変数名・関数名の変更，定数の変更，式の書き換え，コードの分割の剽窃行為によるソースコードの変更には関連ページを検出できなかった．本提案手法では，検索用のキーワードと完全一致の検索方法で関連ページを特定する．名前の変更，コードの分割，式の書き換えなど変更があったら，作成した検索用のキーワードが変わり，検索 API で取得するデータを変わるため，剽窃チェックの結果も変わる．剽窃パターンの順序に依存しない式の入れ替えの結果は，完全コピーと同じ結果があった．このパターンを対応できた理由としては，本提案手法では，解答のソースコードから行単位でコードを抽出するため，行単位のソースコードの入れ替えがあっても，関連ページの検出に影響がないことである．

5.6 考察

筆者は色々な実験を行い，提案手法の有効性と提案手法が対応できる剽窃パターンについて検証した．5.5 の結果をまとめると，以下の結論があった．

結論 1：実験データについての Web 剽窃チェックが行われ，関連ページの検出精度 Precision は 71.4%である．本提案手法で，ある程度 Web 剽窃を検出できる．関連ページが検出されない可能性がある．関連ページが検出されない場合，Web サイトからコピーしていないことを言いきれない．今後，Google，Yahoo!検索 API の利用についての検討が必要だと考えられる．

結論 2：検索用のキーワードが長くなると，関連ページの検出精度が上がる傾向がある．しかし，検索用のキーワードが長すぎると，検索 API でデータを取得できなくなる可能性がある．今後，検索用のキーワードの作成ルールについての検討が必要だと考えられる．

結論 3：同じプログラミング問題の解答について，短時間内，関連ページの検出率が変わらない．解答のソースコード，Web 剽窃チェック用のキーワードなどを変更しない場合，短時間内，再チェックの必要性がない．

結論 4：プログラミング問題の Web 剽窃チェック用のキーワードは剽窃チェックの結果に影響が大きい．関連ページが検出されない場合，Web 剽窃チェック用のキーワードを変更し，再チェックが必要である．また，Web 剽窃チェック用のキーワードを利用するか否かは，剽窃チェックの結果があまり変わらないことを判明した．今後，Web 剽窃チェック用のキーワードを利用しないについての実装が必要である．

結論 5：本提案手法では，ソースコードに定数名・変数名・関数名の変更，定数の変更，式の書き換え，コードの分割などの剽窃行為があった場合，関連ページを検出できない．ソースコードに，順序に依存しない式の入れ替え，インデントの変更，コメントの書き換えの剽窃行為があった場合，提案手法で作成した検索用のキーワードは完全コピーと同じものであるため，関連ページを検出できる．

第6章　おわりに

本プロジェクトでは，「問題解決の実践によるプログラミング教育向け学習支援システム」として，ユーザ管理機能，問題管理機能，授業管理機能，自動評価機能，解答ソースコードの剽窃チェック機能付きの Web アプリケーションを開発した．

本プロジェクトで開発した Web アプリケーションの開発には，提案した機能を実装したが，他の便利な機能の実装はまだ不十分である．筆者が担当した Web 剽窃チェックでは，筆者の提案手法で Web 剽窃チェックの機能を実装した．実験の結果からみると，本提案手法は完全コピー，コメントの書き換え，インデントの変更，順序に依存しない式の入れ替えによるソースコードの変更には関連ページを検出できる．定数名・変数名・関数名の変更，定数の変更，式の書き換え，コードの分割によるソースコードの変更には関連ページを検出できない．

今後の課題として，大きく分けると，2 つの課題がある．

1 つ目は Web アプリケーションの利便性機能を実装する．本システムでは，検索，ソート，ランニングなど便利の機能まだ実装していない．今後は，タスクリストの基に，優先度の高いタスクから実装する必要がある．また，本プロジェクトでは，手動でテストを行った．テストコードはあまり書いていないため，今後，テストコードを充実にする必要もある．

2 つ目は Web 剽窃チェック機能の改善．本システムでは，Microsoft の Bing 検索 API を利用し，関連ページのデータを取得する．しかし，検索エンジンのアルゴリズムによって，検出できない関連ページが存在する．関連ページの検出率を上げるため，Google，Yahoo! 検索 API の利用し，実装する必要がある．また，本提案手法は，検出された関連ページを教員に表示するだけ，教員は関連ページにアクセスしないと，剽窃しているかどうかまだ判断できない．今後，検出された関連ページ中身のソースコードを収集し，解答ソースコードと同じソースコードを載っているかどうかを確認する機能の作成が必要である．

謝辞

研究プロジェクトにおいて，指導教員である筑波大学システム情報系の田中二郎教授，課題担当であるアランニャ・クラウス助教からご支援・ご指導をいただきました．心より感謝いたします．

また，大学院一年生の時の課題指導教員である山戸昭三研究員からは親切なご指導やアドバイスを頂きました．心より感謝いたします．

本プロジェクトのメンバである大桶真宏，申屠偉誠には，プロジェクトを進める上での多くの協力をいただきました．共にプロジェクトを遂行できたことに深く感謝いたします．

最後に，大学でお世話になった全ての方々，学生生活を支えてくれた家族に心より感謝いたします．

参考文献

[1]Programming Challenges,
(http://www.programming-challenges.com/),(参照 2016-1-10).
[2]UVa Online Judge,(https://uva.onlinejudge.org/),(参照 2016-1-10).
[3]Manaba,(https://manaba.tsukuba.ac.jp/ct/home),(参照 2016-1-10).
[4]Moodle,(http://demovpl.dis.ulpgc.es/moodle/),(参照 2016-1-10).
[5]AIZU ONLINE JUDGE,(http://judge.u-aizu.ac.jp/onlinejudge/),(参照 2016-1-10).
[6]東京大学プログラミングコンテスト, (http://www.utpc.jp/),(参照 2016-1-10).
[7]Docker,(https://www.docker.com/),(参照 2016-1-10).
[8]Nginx,(http://nginx.org/),(参照 2016-1-10).
[9]Unicorn,(http://unicorn.bogomips.org/),(参照 2016-1-10).
[10]Sim,(http://dickgrune.com/Programs/similarity_tester/),(参照 2016-1-10).
[11]CloneDigger,(http://clonedigger.sourceforge.net/),(参照 2016-1-10).
[12]Bing,(https://datamarket.azure.com/dataset/bing/search#schema),(参照 2016-1-10).
[13]Taiga,(https://taiga.io/),(参照 2016-1-10).
[14]布目 淳，福澤 理行，平田 博章：プログラミング実習におけるコード評価のための e ラーニングバックエンドシステムの開発，信学技報，Vol.108,No.24,pp.59-64,(2008)
[15]上田 和志，富永 浩之：類似性に基づくレポート剽窃の検出ツールのプログラミング課題への適用，情処研報，Vol.2010-CE-107,No.9,pp.1-8,(2010)
[16]田代 崇，他：Web ページを対象とした著作権違反自動検知システム，データベース・システム研究会報告，情報処理学会，Vol.2006, No.78, pp.27-33, 2006.07
[17]高橋 勇，他：Web サイトからの剽窃レポート発見支援システム，信学論，Vol.J90-D,No.11,pp.2989-2999,(2007)
[18]馬 青，松山 秀人，村田 真樹：Web サイトからの剽窃レポート発見支援システム，研究報告自然言語処理（NL） 2009-NL-194(1), 1-7, 2009-11-09
[19]適合率(Precision,精度)について,
(http://www.cse.kyoto-su.ac.jp/~g0846020/keywords/precision.html),(参照 2016-1-10).