Vol.2010-CE-103 No.4
2010/3/6

類似性に基づくレポート剽窃の検出ツールのプラグイン化

上田 和志, 富永 浩之

香川大学工学部 〒761-0396 香川県高松市林町 2217-20
E-mail: s09t453@stamil.eng.kagawa-u.ac.jp

あらまし レポートのオンライン提出においては，学生が安易に答案の剽窃を行いやすい．そこで，類似性に基づくレポート剽窃の検出ツールを開発し，各種の教育支援システムにプラグインとして提供する．類似性の判定には，編集距離および圧縮比率に基づく 2 つの手法を併用する．適切な事前処理を行って，実用的な判定精度と効率化を目指す．特に，ソースコードの自動採点システムに導入し，不正行為の摘発だけでなく，レポート提出時の警告にも利用する．

キーワード オンラインファイル提出, レポートの剽窃検出, テキストとソースコードの類似性

Plug-in Module of Detection Methods for Plagiarism of Online Reports based on Similarity

Kazushi UETA, Hiroyuki TOMINAGA

Faculty of Engineering, Kagawa University

2217-20 Hayashi-cho, Takamatsu, Kagawa, 761-0396 Japan
E-mail: s09g453@stmail.eng.kagawa-u.ac.jp

Abstract Recently, the network environment in classroom has been much improved. However, the facilities may promote the imprudent plagiarism of other's answer in online report submission. We developed the detection tool for report plagiarism based on similarity. We adopt two kinds of approach for similarity judgment. The one is text base by edit distance. The another is binary base by file compression ratio. We prepare some preprocess for adequate judgment precision and cost performance. We apply it for source codes of C language. It is as a plug-in module of tProgrEss, a support server for programming exercise with a contest style. It uses not only for exposure of the fraudulent activity by a teacher but also for the warnings at the time of students' uploading.

Keyword Online file submission, Detection methods of report plagiarism, Similarity of text and source code

1. はじめに

大学キャンパスのネットワーク環境が整備され，学生 1 人に 1 台の PC を前提とした授業が可能となってきている．さらに，自宅の PC から大学ネットワークにアクセスしたり，ノート PC や携帯端末を持参することが日常的になっている．授業資料の配布，レポートの提出なども，オンラインで行うことが普通になってきた．オンラインでのレポート提出は，答案ファイルの管理や採点業務のシステム支援など，教員側にとって利便性が高い．反面，学生側には，安易なコピーが可能で，不正行為を誘発する原因にもなる．他人の著作物を引用として明示せず，自分のものとして掲載する剽窃は，決して見逃してはならない問題である．

本研究では，レポート同士の類似性を基に，

剽窃行為の検出を行う方法を提案し, 汎用的なツールを開発する[1]．類似性の判定に，編集距離および情報距離に基づく 2 つの手法を用いる．その上で，対象データの特性に応じた事前処理を導入し, 効率性と精度のバランスを考慮して，実用的なツールを目指す．特に，C 言語のソースコードへの適用を検討する. 使いやすいユーザインタフェースを提供し, 教育現場での利便性を高める．また，システムインタフェースを整備し, プラグインモジュールとして, 各種のシステムに組み込めるようにする．

2. レポート剽窃の分析と検出方法

オンラインでのコピーは，PC 上の操作として，コピー・アンド・ペーストで簡単に行えるため,「コピペ」という俗称がある．そのくらい, 多くの学生にとって，日常的に可能な行為となっている. 学生によるレポートの剽窃には, Wikipedia など, Web 上の文献からのコピーと, 学生同士の答案のコピーとがある.

前者の検出については，コピー元の文章は Web 上にあるため, その全てと比較することは困難である．そのため，システムが剽窃されていそうな文章を予め抽出しておかなければならない. 関連研究として，答案の文章の一部を検索エンジンにかけて, コピー元を突き止める方法がある[2]．特に，金沢工業大学知的財産科学研究センター長の杉光一成教授が考案し，株式会社アンクが開発した, コピペ判定支援ソフト「コピペルナー」が販売され，話題となった[3]．判定結果にコピペ割合やコピペ元も表示し，直感的な GUI となっている．

後者については, 全ての答案の組合せに対し, 類似性を調べることになる．この方法は，答案数が多いと，非常に時間のかかる処理になる．「コピペルナー」には，クロスチェッカーとして，こちらの機能も存在する．ただし，製品の説明からは, 日本語文書への適用を想定したものと思われる．いずれの方法でも，システム的な処理だけで, 剽窃かどうかを完全に見分けることは困難である．最終的には，教員による確認が必要である．また，語句の説明，自由な論評，数学の解答，プログラムなど，対象領域によっても，その精度は大きく影響される．文書の類似度を求める手法については，n-gram を用いる方法や, 単語の出現数を比較する方法が提案されている[4].

3. 文字列同士の距離に基づく類似度

3.1. レーベンシュタイン距離

文字列同士の類似度としては, 編集距離に基づく方法がよく知られている．特に，最も簡単なレーベンシュタイン距離が用いられる. これは，文字単位の挿入・変更・削除という基本操作の最低回数で，距離を求めるものである．近年，遺伝子解析などの生物情報学の分野でも，基本的な手法として利用されている．例えば，文字列 bcabc と abdd は，以下の 2 通りの一致のさせ方が考えられるが, 基本操作の回数がより少ない方を採用して距離 4 となる．

a	b	d	d		1 文字一致	距離 5
	b	c	a	b	c	
			a	b	d	d	2 文字一致	距離 4

3.2. レーベンシュタイン距離の効率的な計算

レーベンシュタイン距離は, 漸化式を用いた再帰的定義で与えられる．ただし，文字列長を |X|，先頭文字を削除した文字列を X'で表す.

$$LEV(S,T)=\begin{cases}|S| & ;\ T=\phi \\ |T| & ;\ S=\phi \\ LEV(S',T') & ;\ \text{先頭が一致} \\ \min\begin{Bmatrix}LEV(S',T)\\ LEV(S,T')\\ LEV(S',T')\end{Bmatrix}+1 & ;\ \text{不一致}\end{cases}$$

しかし，この定義に基づく素朴な計算では，再帰呼出が樹状展開される. 文字列長が増えると，膨大な計算量が必要になる．実際には，再

帰呼出で何度も同じ計算をしている．そこで，レーベンシュタイン距離の計算として，動的計画法による効率的な算法が知られている．図 1 のような作業表を作り，左上から右下へ，最も操作回数が少なくなるような経路を選んでいく．初期値として，文字列の先頭に仮想的な空文字を入れておき，最上行と最右列に，0 から順に数値を入れておく．左上隅の 0 から始め，右・下・右下への移動が，横の文字列からみて，削除・挿入・変更に相当し，値を 1 加算する．ただし，文字が一致するときは，右下の移動において加算しない．これらの 3 通りの値のうち，最小となるものを作業表に書き込んでいく．右下隅に達したときの値が，求める距離である．

	ϕ	b	c	a	b	c
ϕ	0	1	2	3	4	5
a	1	1	2	2	3	4
b	2	1	2	3	2	3
d	3	2	2	3	3	3
d	4	3	3	3	4	4

図 1 レーベンシュタイン距離の計算

3.3. レーベンシュタイン距離による類似度

レーベンシュタイン距離は，文字列同士がどれくらい異なるかの絶対量を表しており，文字列長の影響を受ける．一般に，|S|≧|T| ならば |S|－|T|≦LEV(S,T)≦|S| となる．特に，T＝ϕ なら LEV(S,T)＝|S| となる．そこで，編集距離を割合として正規化し，区間[0,1]に収まる類似度 Sim_L を以下のように定義する．

$$Sim_L(S,T) = 1 - \frac{LEV(S,T)}{\max\{|S|,|T|\}}$$

この値は，完全に一致しているとき，1.0 となり，全ての文字が異なるとき，0.0 となる．

3.4. コルモゴロフ記述量による情報距離

文字列同士の類似度として，情報距離を用いる方法も提唱されている．情報距離は，文字列の複雑さを，その文字列を出力するための算法の記述量の最小値と定義するコルモゴロフ記述量に基づく．すなわち，データ圧縮の下限である．これは特定のプログラム言語に依存しない抽象的な定義であるが，通常は 1 つの言語を仮定して議論を進める．例えば，文字列 0123012301230123 は，Ruby 言語では print "0123" *4 と記述できる．ここで，文字列 T を 0123 とすれば，print T *4 とさらに短くできる．コルモゴロフ記述量は K(S)と書く．また，文字列 T に含まれる情報を用いて，文字列 S を圧縮したものを相対コルモゴロフ記述量と呼び，K(S|T)と書く．一般に，K(S)≧K(S|T) となるので，K(S)－K(S|T) は，T を用いたデータ圧縮の改善度，また S に含まれる T の情報量である．これを両者の類似度とみなす．

3.5. NCD による類似度

実際には，全ての記述方法を調べて，コルモゴロフ記述量を正確に求めることは不可能である．そこで，コルモゴロフ記述量を具体的な圧縮算法で近似し，距離として定義したものが正規化圧縮距離 NCD(Normalized Compression Distance)である[5]．NCD は，汎用的なデータ間の類似度の算出方法として，音楽や文学，遺伝子情報など，様々な分野で応用されている．

実用的には，一般的に用いられる圧縮ツールで圧縮したサイズ C(S)で十分である．相対記述量については，文字列 T,S を連結した文字列 TS を使い，C(S|T)＝C(TS)－C(S) で代用する．この値は，文字列同士がどれくらい異なるかを表しているが，文字列長の影響を受ける．そこで，割合として正規化し，NCD を以下のように定義する．

$$NCD(S,T) = \frac{C(TS) - C(T)}{C(S)} \quad ; \quad C(S) \geq C(T)$$

NCD が 1.0 のときに 2 つの文字列は全く異なり、0.0 のときは完全に一致しているといえる。類似度としては, 1.0 が完全一致となるように, 1 から引いて，以下のように Sim_N を定義する．ただし，実際の圧縮ツールでは，ヘッダ情報などが付加されるため，正確に 0.0 から 1.0 の間に収まるわけではないが, 実用的に問題はない.

$$Sim_N(S,T) = 1 - \frac{C(TS) - C(T)}{C(S)} = \frac{C(S) + C(T) - C(TS)}{\max\{C(S), C(T)\}}$$

4. レポート剽窃の検出方法と事前処理

4.1. 編集距離の計算量の低減

レーベンシュタイン距離の計算は, 動的計画法を用いても, 文字数の 2 乗に比例する処理時間が必要である. レポートの文章が長くなると膨大な計算時間がかかる．しかし，各文字の比較は，一致/不一致の単純な判定である．したがって，比較対象の文字種を増やし，文字数を削減することが考えられる．

そこで，事前処理として，簡易な形態素解析を施し，単語単位に分割して，各単語を 1 つの文字として照合を行う．その際，各単語をハッシュ値に置き換え, 整数列に対する処理として実装する．また,「て，に，を，は」などの助詞や助動詞など, 内容に影響しない機能語を除外する. また, 指定した様式に基づく文書では, 共通のテンプレート部分も除外しておく.

4.2. 文書サイズの差による影響

一般に，レポートは，同じ問題に対する解答である. 記述に必要な語句は重複するのが当然であり，類似度がほぼ 0.0 になることはあり得ず，ある程度の値以上となる．また，比較する一方の文書が非常に短い場合, 編集距離が長い方の文字列長にほぼ等しくなるので, 類似度は 0.0 に近くなる．情報距離についても，短い方の文書が圧縮にほとんど影響を与えないので, 同様に 0.0 に近くなる．剽窃の有無という観点からは，見かけの距離が大きく見積もられ，類似度が不適切な値となっている．そこで，データサイズによって, 幾つかのグループに分けることが必要である.

4.3. 編集距離に基づく類似性の事前処理

編集距離に基づく類似性の計算において, テキスト文書に対する事前処理を述べる．まず，適用範囲を広げるため，PDF や MS Word などのバイナリ文書からも文字列データを抽出し，テキストファイルを生成しておく. 図版は削除し，表組から文字列を取り出し，箇条書きはそのままとする．この部分の処理は，テキスト抽出モジュールとして独立化させておき, 必要に応じて適切な方法を採用する．次に，日本語文書の場合，形態素解析を行い，単語単位に分割する．さらに，機能語を削除し，自立語のみを残す. 英文の場合は，分かち書きの各語をそのまま単語とみなす.

次に，ハッシュ化を行う．各単語は 2 文字 4 バイトずつで区切り, UTF-8 の文字コードからなるビット列として XOR でハッシュ化する．これにより, 長い単語も 4 バイトに正規化され, char 型での文字照合から，unsigned int 型による単語照合に帰着させることができる. 例えば,「知識工学基礎」は, 知識・工学・基礎と分け，各 2 文字の文字コードを 32 ビット符号無整数とみなし，ビット単位の XOR をかける．ハッシュ空間は約 42 億であり，十分な大きさである．大学生の標準的な語彙数は，多い人でも 6 万程度と言われており, ハッシュキーの衝突は起こりにくい.

4.4. ソースコードに対する事前処理

ソースコードの事前処理としては, 字句解析によってトークンに分け, 予約語やユーザ定義の識別子のみを抽出する. 括弧やコンマなどの区切り文字は, トークン数としてはかなりの量になる．しかし，構文的にほぼ正しいコードの

場合, 類似性に大きな影響を与えないと考えられる．また，コードの一部を補完するような問題では，共通のコード部分を除外しておく．

コード自体を簡易パーサにかける以外に, アセンブラ言語の中間コードに変換し, そのコード上でのトークン列とする方法が考えられる．この方法では, 変数名の置換や while 構文を for 構文に書き換えただけといった, 軽微な変更の影響を受けない．データ量は余り減らないが, より精密に類似度を算出でき, 少し細工された剽窃も見つけられる. コード自体のトークン列で剽窃の候補を絞り, 中間コードのトークン列でさらに確認するという使い方が考えられる．

5. レポート剽窃の検出ツールのプラグイン化

5.1. 検出ツールの実装

事前処理の主要部分および剽窃検出の本体処理は，Ruby 言語で実装した．また，Web アプリケーションからのプラグイン利用を想定して，インタフェースを整備する．ただし，編集距離は計算時間がかかるため, その部分だけを C 言語で実装し，Ruby スクリプトから呼び出している．形態素解析ツールには，MeCab を用いる[6]. アセンブラには, GCC を用いる．

圧縮ツールとして, gzip と bzip2 を検討する．gzip は圧縮率が低いが速い，bzip2 は圧縮率が高いが遅い，という特徴がある．圧縮率が高いアルゴリズムを利用すれば, 類似度の精度を上げることができる．しかし，テキストデータに対しては, gzip で十分な圧縮率を得ることができるため，テキストデータには gzip，バイナリデータには bzip2 を使用する．

5.2. 検出ツールのモジュール構成

レポート剽窃の検出ツールは, 編集距離と情報距離のそれぞれで, 図 2 のようなフィルタ群として設計する．対象領域の特性に応じて，両者の手法を組み合わせ, パラメタ設定が柔軟に行えるようにする．また，ツールだけで，剽窃を検出するわけではないので, 判定結果や疑わしい組を，教師に分かりやすく表示する GUI も必要となる．単独で用いる場合には，対象データを指定し，全ての組合せの類似度を求め, その順位表を表示する GUI を用意する．

5.3. WebBinder へのプラグイン

本研究室では, 学内サイトの授業支援サービスと連携し, 授業関連の個人情報を集約して提示する学生ポータルサイト PASPort を提案している(図 3)[7]．そのドキュメント管理サービスとして，レポート作業を支援する WebBinder を開発中である(図 4)[8].

WebBinder は，オンラインストレイジの拡張であり, 教員サイトおよびファイル共有サーバとユーザの間で橋渡しの役割をする．学生は, 学内外を問わずレポート課題にアクセスできる. 文書作成などのアプリケーション依存の作業はローカル側で行い, 関連ファイルの保管はサーバ側で行う．取得・作成・提出の 3 段階で支援機能を提供する．特に，問題選択の条件が煩雑で, 複数の成果ファイルを要求されるプログラミング課題の提出を支援する．

レポート剽窃の検出ツールは，教員側での不正行為の摘発のため，WebBinder の教員サイトへのサービスとして，API を提供する(図 5). また，提出フェーズにおいて，提出前の学生への警告としても利用する．

5.4. tProgrEss へのプラグイン

本研究室では, 小コンテスト形式の C プログラミングの演習支援サーバ tProgrEss を開発している[9]．プログラミング問題の解答となるソースコードをアップロードし, サーバ側でコンパイルして, 入力サンプルを与えて実行する．実行結果と出力サンプルを照合し, プログラムの正誤判定を行う．出力サンプルは，正解となる模範プログラムの実行結果である. 正誤判定

には，6 段階あり，最初の段階で，不正提出の判定を行っている．空ファイル，巨大ファイル，バイナリなどを排除する．また，プロセス制御文など悪意のある危険コードのうち，字面で判断できるものも排除する．

剽窃検出は，上記の不正提出の判定部分に含める．既に提出された他の学生のコードと明らかに類似性の高い場合を排除する．疑わしいが明らかと言えない場合は，警告のみを出して，以降の正誤判定を進める．コードにフラグを立てておき，後で教員が目視で確認する．なお，同じ学生の修正されたコードとの比較は行わない．tProgrEss は，Ruby 言語で開発されており，Ruby 言語で実装された剽窃検出ツールとの親和性が高い．即時的な判定が求められるため，効率性を高める事前処理が重要となる．

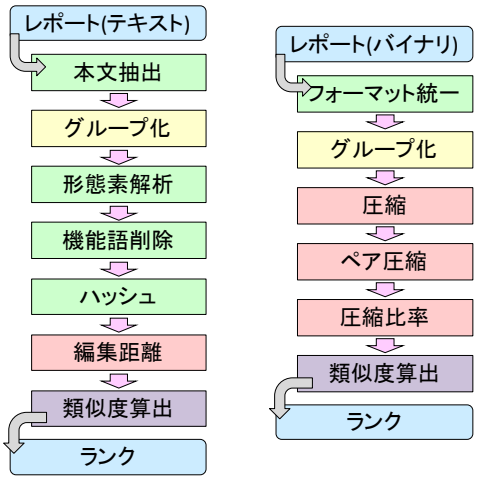

図 2 剽窃検出ツールのモジュール構成

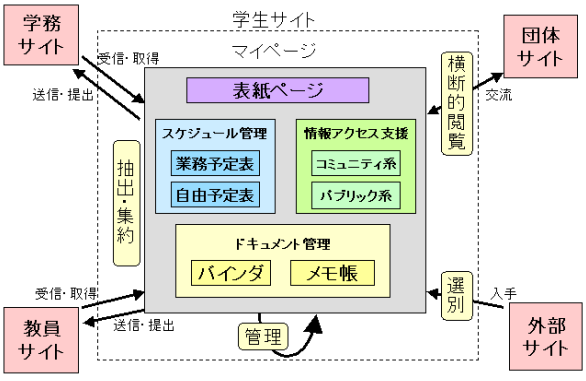

図 3 PASPort のシステム構成

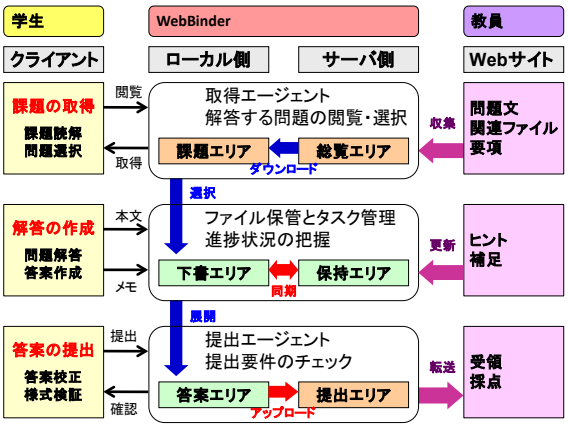

図 4 レポート作業支援システム WebBinder

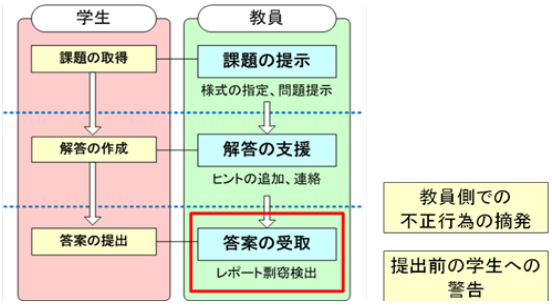

図 5 レポート剽窃の検出ツールの組込み

6. ソースコードに対する剽窃検出の実験

6.1. 実験対象のソースコード

C 言語のソースコードに対して，剽窃検出の予備実験を行った．対象データは，2004 年度から 2008 年度までの「ソフトウェア開発演習」(3 年次選択科目)におけるプログラミング課題のソースコードである．内容は，知識情報処理の分野の応用であり，ポーカーにおいて捨てる札を選ぶ戦略を関数 strategy() として実装するものである．この関数を記述するファイルは，ゲーム進行の共通処理とは別ファイルになっており，学生に提示したコードは，関数原型と僅かなコードの例示だけである．そのため，純粋に類似度を判定するのに適している．データ数は 163 件で，類似度を計算する組としては，13203 件となる．平均で，ファイルサイズは 10212 バイト，コメントを除外して，行数は 273 行，文字数は 4609 字である．予約語と識別子

のトークン数の平均は，元のコードで 536，中間コードで 3274 であり，前者は後者の約 1/6 である．また，GZIP による圧縮では，平均 2401 バイトとなっており，約 1/4 に圧縮されている．

6.2. ソースコードに対する処理時間

編集距離による類似度の計算では，事前処理として字句解析に 1 秒以下，本体処理に約 4 分かかった．1 件のデータを他の 162 件の対象データと比較する時間は，1.5 秒程度である．1 組だけの比較は，0.01 秒程度である．中間コードを用いた場合は，事前処理のアセンブルに 6 秒，本体処理に約 50 分かかった．ただし，構文エラーなどでアセンブルできないコードもあったため，正しく中間コードが生成できたのは，138 件である．1 件につき，比較の総時間は，18 秒程度である．いずれも，予備実験で C 言語で完全に記述していたときの 3 倍の時間を要している[1]．情報距離による類似度の計算では，事前処理として GZIP での圧縮に 2 秒，本体処理に 23 秒かかった．1 件につき，比較の総時間は，0.14 秒程度である．実験に用いたマシンの性能は，CPU Intel Core2Duo 3.0GHz, RAM DDR2 4GB である．OS は，Linux の 64bit 版 CentOS 5 上である．

6.3. ソースコードに対する実験結果

ソースコードについては，元コードおよび中間コードについて，編集距離による類似度の相関が高かった．両者の値が高い組を目視で確認した結果，上位から 5 件が明白な剽窃であると判断できた．このうち，2 組は学年を超えた学生のものであり，両者の所属ゼミが一致していることから，剽窃の疑いが極めて濃厚であるといえる．

一方，情報距離による類似度は，編集距離による類似度との相関が，必ずしも高いと言えなかった．類似度が高い組には，明らかな剽窃が認められないものが含まれていた．その理由として，例示のコードから，ほとんど修正していないコードや，if 構文を延々と繰り返す冗長なコードの存在が挙げられる．前者は，剽窃以前の状態である．後者は，圧縮比が高く，自己コピペ状態になっていた．これらのコードは，剽窃検出としては除外するが，別の観点から教育的指導を行うべきである．

6.4. 検出方法の精密化

検出手法の精密化について，検討している手法を述べる．まず，オリジナル部分の多少を測る指標として，非対称な依存度を導入する．X の Y に対する依存度は，以下で定義する．類似度の性質から依存度も区間[0,1]に収まる．

$$Dep(X,Y)=\begin{cases} Sim(X,Y) & ; \ |X| \geq |Y| \\ \dfrac{|Y|}{|X|} Sim(X,Y) & ; \ |X| < |Y| \end{cases}$$

例えば，S=abcdefgh, T=abx のとき，類似度は Sim(S,T)=6 で，両者の依存度は Dep(S,T)=0.25, Dep(T,S)=0.25×8/3＝0.67 となる．S は T と似ている部分 ab があるものの，cdefgh とオリジナルな部分の方が多い．T は S にも含まれる ab を除けば，オリジナルな部分は x しかない．

この依存度の対象データ全てに亘る総和として，剽窃度 Plag(S)＝ΣDep(S,X) を定義する．剽窃度が大きいと，他と似ている部分が多く，オリジナルな部分が少ない．すなわち，コピペの疑いが高い．特に，サイズが小さく，他からつまみ食いしたようなデータを検出し易い．

7. おわりに

レポート剽窃の効率的な検出方法を検討し，実用的なツールを開発中である．類似性の判定には，編集距離および情報距離に基づく 2 つの手法を併用する．適切な事前処理を行って，実用的な判定精度と効率化を目指す．特に，C 言語のソースコードに対しては，字句解析によるトークン列への帰着，共通コードや区切記号の

削除などを行う. アセンブルした中間コードも利用する．Ruby 言語で実装し，ユーザインタフェースとシステムインタフェースを整備する. レポート作業支援 WebBinder やコード正誤判定 tProgrEss にプラグインとして組み込む. 実験では，編集距離による類似度では，明らかな剽窃検出に成功した. 情報距離による類似度は，コードの特性から，ノイズとなる候補が現れた. どちらも場合も, 精度を高めるためには, 小サイズのデータを除外しておく必要がある. また，Ruby 言語での実装では，処理時間が多くなるため, C 言語での記述部分を増やす必要がある．これらの問題点を改良し，実際の運用を目指す.

文　献

[1] 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開発", 信学技報, Vol.109, No.335, pp.61-66, (2009).

[2] 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).

[3] アンク社, コピペ判定支援ソフト コピペルナー, http://www.ank.co.jp/works/products/copypelna/.

[4] 深谷亮, 他, "単語の頻度統計を用いた文章の類似性の定量化 – 部分的類似性の考慮 –", 信学論, Vol.J87-D-II, No.2, pp.661-672, (2004).

[5] Sven Helmer. "Measuring the Structural Similarity of Semistructured Documents Using Entropy", Proceedings of the 33rd international Conference on Very Large Data Bases, pp.1022-1032, (2007).

[6] 形態素解析エンジン MeCab, http://mecab.sourceforge.net/.

[7] 三嶋宏資, 福島直文, 富永浩之, "授業支援サービスと連携した個人適応の学生ポータルサイト – 授業関連情報を時間割ベースで共有する予定表 –", 信学技報, Vol.107, No.536, pp.101-106, (2008).

[8] 上田和志, 富永浩之, "プログラミング課題のレポート提出を支援するオンラインストレイジ WebBinder の開発", JSiSE 研究報告, Vol.23, No.6, pp.112-119, (2009).

[9] 川崎慎一郎, 富永浩之, "競争型学習を取り入れた入門的 C プログラミング演習 – 演習支援サーバ tProgrEss の出題解答と採点結果のページ表示の改良 –", 信学技報, Vol.109, No.335, pp.187-192, (2009)