Webページに含まれる類似プログラム断片の検索手法の提案

M2011MM072 立道昂太
指導教員：吉田敦

1　はじめに

リポジトリサイトや解説サイトなどのプログラムを掲載したWebページが，プログラム開発や学習のために広く利用されている．これらは，開発に用いるAPIやアルゴリズムの現実的な使い方を知るだけでなく，再利用により開発効率を高める効果もある．

閲覧者は閲覧しているページでは足りない情報を得るために，類似するプログラム断片を持つWebページを参考にするが，その検索は短時間では実現できない．閲覧者が着目しているプログラムは，関数や複数の文の並びなど様々な粒度のプログラムの断片である．このプログラム断片と類似するプログラムを検索する方法としてコードクローン検索ツール [3] を応用する方法もあるが，あらかじめ検索候補の全プログラム間での類似性を求める必要がある．しかし，閲覧者が着目するプログラムをあらかじめ予測して解析対象に含めることはできず，検索に時間をかけて類似性の解析をする必要がある．

本研究は，類似するプログラム断片を掲載したWebページ検索の手間の削減を目的とし，任意のプログラム断片に対して，短時間で類似断片を含むWebページを検索する手法を提案する．

類似プログラムを特定するために，検索前にどのような情報を構成するかが課題である．あらかじめ検索前に特定のための情報を整理し利用することで，検索時の時間を短縮できる．しかし，断片の粒度が異なるので，コードクローン探索や，プログラム部品のメトリクスを求めてメトリクスをキーとして検索する手法 [7] のように，あらかじめ類似関係を分析した結果を用意できない．そこで，プログラムを抽象化した各文を単位としたインデックスを作成し，検索時に類似度の高い断片の範囲を特定する手法を提案する．

抽象化では，各文ごとに意味に関係ない字句や重複した字句を取り除き，残りの字句を並べ替える．本研究ではこれを「簡素化」と呼び，意味は同じだが，字句の出現順序が異なるなどの表現の違いを吸収した統一表現に変換する．文を単位とする理由は，断片を理解するときの処理の基本単位となるからである．

簡素化で表現の違いは吸収できるが，類似した記述が同じ表現になるとは限らない．そこで，自然言語の検索手法で用いられるN-gramを応用し，部分的に一致する箇所を特定して，その組み合わせから類似断片を求める．具体的には，簡素化後の字句から，字句を単位としたN個の字句列を生成し，その字句列が多く適合した文を類似文とみなし，類似文が連続する箇所を類似断片として特定する．

提案手法では，検索対象の情報をあらかじめ蓄積することから，最新のWebページの内容にずれが生じる可能性がある．そこで，特定したプログラムの記述からパターン記述を作成し，ページ内でパターンに一致する箇所を特定することで，記述の更新に対応する．この手法を利用して類似プログラム検索方法を提案し，実際の検索システムと検索時間，精度を比較することで提案方法の有用性を示す．本研究ではプログラミング言語として実際の開発にも用いられ，多くの解説サイトが存在するC言語を対象とする．

2　関連研究

類似するプログラムの特定方法としてSPARS-J[7] の利用があるが，登録したプログラム断片と検索対象のプログラム断片の粒度が一致するとは限らないので，そのまま適用できない．検索時に，プログラム断片の粒度と同じ粒度に検索対象のWebページのプログラムを切り出して，類似を判定することで，断片の特定に応用できる．しかし，これは検索時に改めてメトリクスを求める必要があり，短時間で実現できない．

プログラムの検索方法としてKoders[2] やKrugle[1] などのソースコード検索エンジンの利用があるが，検索クエリの表現力が低く，検索結果として目的と合わないものもあらわれる．ゆえに，検索結果の一覧から目的のプログラム記述を含むWebページを探索する手間がかかるので，短時間で特定するという要求を満たさない．

3　プログラムの類似判定

類似した断片は，類似する字句列を持つ文が連続する箇所として求める．ただし，文には包含関係があるので，単純に文を単位とすると字句の重複が生じる．そこで，以下に示す「文要素」を定義し，この文要素を単位として求める．

> ラベル文．式文．複合文．選択文．繰り返し文．ジャンプ文．宣言．制御文は3つ (制御文開始の文要素，包含された文，制御文終了の文要素) に分割．複合文は3つの文要素 (開始の文要素，包含された文，終了の文要素) に分割．条件文if文のelse字句．

条件文if文のelse字句が分要素である理由は，制御文の一つであるif文にelseが存在した場合，定義に従い3つに分割しても他の文を包含する構造が残るからである．

簡素化は，表現の違いを吸収するために，意味に関係しない要素の削除や要素の並べ替えをおこなうことをいう．これにより，プログラム間で一致する字句列を増やす．類似していないプログラム間でも字句列が一致する可能性があるが，その分，類似しているプログラム間で一致する字句列がより多く現われる可能性がある．また，同じ意味のプログラムがすべて同じ表現になるとは限らないが，簡素化によってその記述の前後の字句列が一致することで，特定ができる可能性がある．

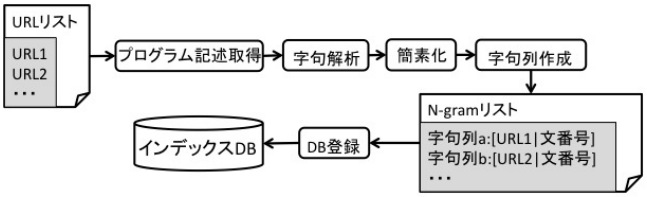

図 1 インデックス DB 構築の処理の流れ

簡素化した断片と検索対象のプログラムそれぞれに対して，字句を単位とした N-gram を利用して字句列を作成し，断片の字句列と一致した検索対象の字句列を特定する．特定した字句列が多く適合している文要素が，断片に類似する文要素であると判断する．

断片と類似するとして特定された文要素のリストは，文要素の出現順に並べ替えて，連続で存在している箇所を類似断片として特定する．そのために，検索対象の文要素はその出現順に番号を付けておく必要がある．また，類似断片を特定する際に，文要素が重複して存在した場合，その重複数をカウントする．これは重複が多いほど，その断片は他の断片候補よりも類似している可能性が高いからである．

4 類似するプログラムの検索方法

類似プログラムを検索するには，まず，検索対象となる Web ページ上のプログラムを収集し，検索に用いるインデックスを作成しておく．検索時には，キーとなる断片と類似する箇所をインデックスから特定する．

4.1 インデックスの構築

閲覧者は，検索を行う前に検索対象の Web ページのインデックスを作成する．インデックスは，N-gram に基づく字句列をキー，字句列が存在する Web ページ上の位置情報で構成される．位置情報は，URL と文番号で構成される．URL は，検索対象の Web ページは複数のドメインからなるので位置情報に含む．文番号は，Web ページごとに文要素の出現順に番号をつけたものであり，連続する文要素を特定するために並べ替える必要があるので位置情報に含む．

処理の流れを図 1 に示す．プログラム記述の取得では，入力された各 URL ごとに，プログラム記述が存在する箇所を特定する方法 [6] を利用してプログラム断片を抽出する．簡素化と字句列の作成の処理は 4.2.1 節で示す．字句列とその位置情報を組としてインデックスを作成し，そのリストをデータベースに登録する．同じ字句列でも位置情報が異なるインデックスは，複数の位置情報を要素として持たせる．

4.2 類似するプログラムの検索手順

検索時は，閲覧者が着目する断片を入力し，インデックス DB を利用することで類似断片を特定する．実際に類似断片特定の処理の流れを図 2 に示す．以降では，提案手法を実行の流れに沿って具体的に記述する．

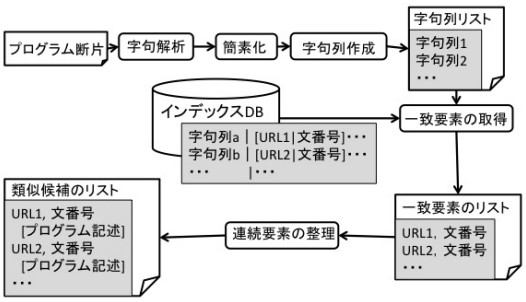

図 2 類似プログラム検索の処理の流れ

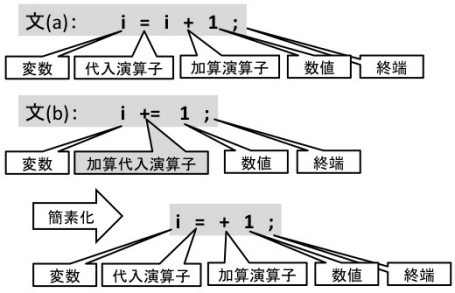

図 3 文の簡素化

4.2.1 簡素化と字句列の作成

文要素の簡素化では，まず，プログラム断片を解析し，文要素に分ける．次に，各文要素ごとに，不要な字句の削除，複数の意味を持つ字句の分解，字句の並び換え，重複する字句の削除をおこなう．不要な字句の削除は，空白やコメントなどプログラムの処理に関係のない字句を削除するためにおこなう．複数の意味を持つ字句の分解は，一つの演算子で複数の意味を持つ演算子をそれぞれの意味を表す演算子に分解する．例えば，加算代入演算子ならば，加算演算子と代入演算子に分解する．字句の並べ替えは，出現順に関係なく類似している文要素を特定するためにおこなう．重複する字句の削除は，文要素内で同一字句が複数存在する場合に，同じ字句が一つになるまで削除する．簡素化されたプログラム断片は，一定の字句数で切り出して字句列を作成する．

具体的な例を図 3 に示す．図 3 の文 (a)，文 (b) は表現は異なるが，ともに同じ処理を意味する文である．これらは，字句やその種類が似ているにも関わらず，一致する字句列が存在しない．文 (a) の中で重複する変数を削除，文 (b) の複数の意味を持つ演算子を意味ごとの演算子に分解し，それぞれを並べ替えることで，二つの文の間で共通する字句列が得られる．

4.2.2 一致要素の取得と連続要素の整理

検索時に，プログラム断片の字句列と一致するキーの要素である位置情報を，インデックスを保存したデータベースであるインデックス DB から取得する．取得の方法を図 4 に示す．

得られた位置情報のリストから，類似プログラム記述の候補箇所を特定する．具体的な方法の適用例を図 5 に示す．リストの先頭要素と次の要素を比較し，URL が一

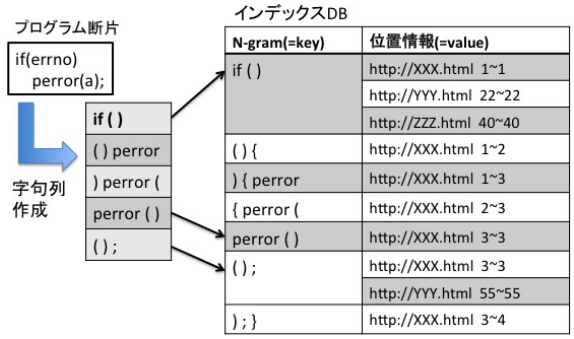

図 4　一致要素の取得

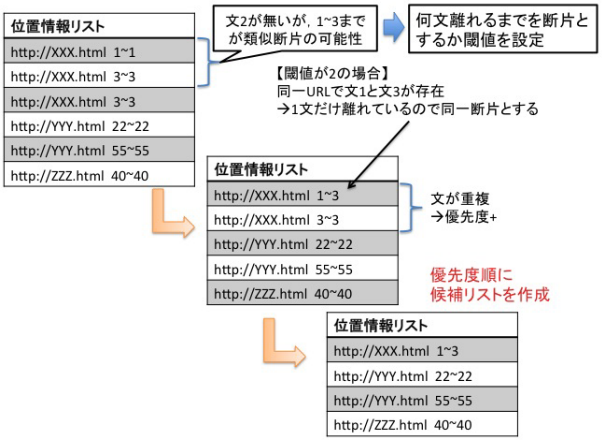

図 5　連続する文番号の整理

致して文番号が連続していたら位置情報を結合する．比較の際に位置情報が重複していた場合は，その重複数をカウントし位置情報を結合する．この重複数が多ければ多いほど，その文中に検索キーの断片の字句列と一致するインデックスの字句列が存在しているので，より類似していると判定する．同じ URL でも文番号が離れている場合，類似箇所が複数あると考えられるが，離れている文が 1，2 文程度ならば，同一の断片を構成している可能性がある．離れていても同一の断片とみなす距離の閾値を設定し，その距離以内であれば位置情報を結合する．また，最終的に得られたリストを，重複数を降順でソートすることで，類似する可能性が高い検索対象のプログラム記述順に並べる．

4.3　更新された Web ページへの対応

インデックスを構築してから，検索時に利用するまでに時間がたって，最新の Web ページの情報とインデックスに差異が生じる問題がある．Web ページのプログラム記述が変更された場合，文番号が変わることで特定した記述と異なる範囲を類似記述として特定する可能性がある．そこで，特定したプログラム記述からパターン記述を作成し，Web ページ内のそのパターンに一致する箇所を特定する [5] ことで，記述の変更に対応する．抽象的なパターンは URL のクエリパラメータとして記述し，Web ページを表示した際に，パラメータを受け取ってページ内を検索する処理を実行することで特定を行う．

5　評価・考察

3 節で提案した類似プログラムの特定方法を実装し，プログラム特定にかかる検索速度とプログラム特定の精度について評価する．

5.1　検索速度の評価・考察

比較対象のシステムはすべて外部のサーバ上に存在しており，単純に検索速度の比較ができない．よって，SPARS-J で使われているメトリクスに基づいた検索方法と，提案した検索方法のアルゴリズムの計算量を比較する．どちらの方法も字句解析をおこない，かつ，最終的にソーティングして結果を出力している．それらの処理を除く，候補箇所を絞り込む処理の計算量を比較する．

メトリクスを利用する検索方法は，検索ワードのプログラム断片が n 個の文で構成されているとすると，メトリクスを求めるための連続する文の組み合わせは n^2 通りであり，メトリクスを求める時間を $O(n)$ とすると，候補箇所を検索する処理を $O(n^3)$ で実現できる．一方，本研究の類似プログラム特定は，プログラム断片に対する簡素化，字句列の作成の処理はそれぞれ $O(n)$ で実現できる．また，データベースはハッシュ表と同様のキーバリュー型の構造をしていることから，各値を取得するのは $O(1)$ で実現でき，候補を検索する時間は $O(n)$ であり，メトリクスを利用するよりも高速で候補が検索可能である．

データベースの登録の計算量を考えると，メトリクスの場合は，検索対象のファイルが p 個それぞれに m 個の文がありメトリクスを求める時間を $O(m)$ とすると，登録の時間は $O(m^3p)$ である．本研究は，一つのファイルは m 個の文で構成される p 個のファイルのインデックスが登録されており，ファイルあたりの字句の数を t として N-gram を作成すると，インデックスは Nt 個得られる．各インデックスが全体として mp 個の文のうち $\frac{1}{q}$ の確率で一致すると仮定すると，インデックス DB 内の位置情報は合計 $Nt\frac{mp}{q}$ 個存在する．ここで，N と q は定数とみなせ，$t = O(m)$ とすると，登録の時間は $O(m^2p)$ である．ゆえに，データベースに登録する処理もメトリクスの場合と比べて高速で実現できる．

5.2　プログラム特定の精度の評価・考察

一般的に，検索システムの性能評価には再現率と適合率が利用される．再現率は，データベース内で検索要求を満たすすべてのファイルをカウントする必要があるが，比較対象のシステムが利用するデータベース内には莫大な数のファイルがあり，一つずつファイルを確認することは困難である．本研究では，再現率を求める代わりに，SPARS-J の評価と同様に適合率と検索システムの比較実験の評価に利用されている ndpm 法を利用する [4]．ndpm 法は文書間を比較し，ユーザにとってどちらが適合している文書であるか判断して評価するものである．ndpm 法を用いて得られる ndpm 値が 0 に近いほど，ユーザの要求を満たした検索結果が上位に出現することを表す．また，検索結果の上位に求める結果が出力されるかが重要である [4] ので，検索結果の上位 20 件を対象とする．プ

表 1 適合率と ndpm 値

N	閾値	適合率	ndpm 値
2	0	0.15000	0.00526
	3	0.15000	0.04733
3	0	0.15000	0.00526
	3	0.15000	0.00526
4	0	0.15000	0.00263
	3	0.15000	0.00263
5	0	0.15000	0.00263
	3	0.15000	0.00263
SPARS-J		0.05000	0.04210
Koders		0	-
Krugle		0	-

ログラム断片と出力結果の類似プログラム候補が類似しているかどうかの判定は，目視でおこなう．

提案した方法は，パラメータによって結果が異なる可能性があるので，N-gram の字句数は 2 から 5 まで，かつ，同一の断片とみなす距離の閾値は 0 から 3 までの合計 16 パターンそれぞれに対して評価する．検索対象は，ドメインの異なる 15 種類の Web サイト合計 130 個の Web ページである．テストケースとして，オープンソースで利用され，解説するサイトも多く存在するアルゴリズムであるバブルソートを実現する断片を対象し，検索した場合の適合率と ndpm 値の一部抜粋した結果を表 1 に示す．

類似プログラム特定方法の入力としてプログラム断片，既存研究のシステムには入力として断片を構成する字句を与えた場合，対象とした 3 つのテストケースすべてにおいて既存研究よりも低い ndpm 値を得た．特に表 1 に示したテストケースにおいては，既存研究に比べて本研究の方が適合率，ndpm 値が共に上回った．このことから，文番号が同じ字句列の重複数が大きいほど類似しているという戦略が正しかったことが分かる．次に，既存研究と比べて適合率も大きいことが分かる．入力として断片を構成する字句を与えたが，既存研究はその字句だけでは特定ができないことからこの結果となった．

パラメータを変更して実験した結果，N-gram の字句数が 3，また閾値は 1 のときが適合率が高く，ndpm 値が低いことが分かった．N-gram の字句数が多い場合，類似プログラムを含む Web ページが検索結果として得られなかった．これは，字句列が長いことから，検索キーと一致する字句列が極端に少なくなることが原因である．閾値が小さい場合，類似断片の特定ができなかった．これは，閾値が低くなることで断片が小さくなり，マッチしなかったと考えられる．逆に閾値が大きすぎると，断片を構成する文の数が多くなり，特定した断片内に類似プログラム以外の記述が多く含まれる．ゆえに，今回の検証結果からは，N-gram は 3，閾値は 1 が妥当であると考察する．

6 おわりに

本研究は，Web ページ上のプログラム断片に関連するプログラムを掲載した Web ページの検索の手間の削減を目的とし，任意のプログラム断片に対して，短時間で類似断片を含む Web ページを検索する手法を提案した．検索技術である N-gram を応用して字句からインデックスを作成し，文に対して簡素化の処理を実現することで，類似するプログラム断片の特定を可能にした．インデックスと最新の Web ページの内容にずれが生じる問題に対しては，特定したプログラム断片からパターン記述を作成し，ページ内でパターンに一致する箇所を特定することで対応した．実際に他の検索システムと評価し，計算量としてはメトリクスを利用するよりも速いアルゴリズムを提案した．また，適合率と ndpm 値を他のシステムと比較評価し，特に ndpm 値が他のシステムよりも本研究の方が良い結果であったことから，順位付けの方法が有益であることを証明した．

今後の課題として，類似プログラム特定の精度向上のために，簡素化の処理として本研究で提案した処理以外にどのような処理を行うべきかを考える必要がある．また，検索対象の Web ページの URL が変更された場合の対象方法も考える必要がある．

参考文献

[1] Aragon Consulting Group, “krugle - software development productivity,” http://www.krugle.com/, 2012.

[2] Black Duck Software, “Ohloh Code Search,” http://code.ohloh.net/, 2012.

[3] Toshihiro Kamiya, Shinji Kusumoto and Katsuro Inoue, “CCFinder: A Multi-Linguistic Token-based Code Clone Detection System for Large Scale Source Code,” IEEE Trans. Software Engineering, vol. 28, no. 7, pp. 654–670, 2002.

[4] 梅森文彰，西秀雄，横森励士，山本哲男，松下誠，楠本真二，井上克郎，“Java を対象としたソフトウェア部品検索システム SPARS-J の実験的評価，”電子情報通信学会技術研究報告，SS2003-49，Vol．103，No．708，pp.19–24，2004.

[5] 清水優，鈴木貴昭，“構文的制約を利用した類似コード断片の検索方法に関する研究，” 2011 年度 南山大学 数理情報学部卒業論文，2012.

[6] 立道昂太，吉田敦，蜂巣吉成，張漢明，野呂昌満，“Web ページ記述内のプログラム断片に対する DOM tree を用いた構文木の構成手法，” ソフトウェアエンジニアリングシンポジウム 2012 論文集，2012 巻，pp.1–6，2012.

[7] 横森励士，梅森文彰，西秀雄，山本哲男，松下誠，楠本真二，井上克郎，“Java ソフトウェア部品検索システム SPARS-J，” 電子情報通信学会論文誌，Vol．J87-D-I(12)，pp. 1060–1068，2004.