

Web 上の教材ファイルからのサンプルプログラムデータベースの構築

杉浦 秀樹[†] 杉本 徹[‡]芝浦工業大学大学院 理工学研究科[†]芝浦工業大学 工学部[‡]

1. 研究の背景と目的

プログラミングを学習する際、教科書を使用することが一般的である。しかし、教科書に掲載されているサンプルプログラムは数に限りがある。

そこで Web 上からプログラムを収集してデータベースを構築し、必要な時に言葉で検索できるようにする。そうすることによって、学習者が多数のサンプルプログラムを閲覧できるようにする。

本研究では、Web 上の教材ファイルからプログラムとその前後に書かれたプログラムに関する文章を抽出する方法を提案する。

2. サンプルプログラムデータベース

2. 1. データベースの構成

サンプルプログラムデータベースは以下の内容から構成される。

- 教材ファイルから抽出したプログラム
- プログラムに関する文章
- 教材ファイルの URL

ここで教材ファイルとは大学の講義資料で、情報工学分野を扱っている PDF やパワーポイント形式のファイルのことである。本研究ではプログラムとして、C 言語で書かれたプログラムのみを対象とする。また、文法的に正しく、関数定義を 1 つ以上含んでいることをプログラムの抽出の条件とする。

プログラムに関する文章としては、そのプログラムで学べる学習項目を表す文（例：「C 言語での文字列の取り扱い」）と、そのプログラムの内容を説明する文（例：「1 から入力した値までの整数を順次表示するプログラム」）の 2 種類を考える。

2. 2. データベースの構築方法

本研究では、以下の手順でサンプルプログラムデータベースを構築する。

- (1) 教材ファイルからのプログラムの抽出
- (2) 抽出したプログラムに関する文章の抽出

教材ファイルは中森[1]が収集したものを使用する。このファイルは検索エンジンに情報工学分野の講義科目名を検索語として与えて集めたものである。

3. 教材ファイルからのプログラムの抽出

3. 1. 教材ファイル中のプログラムの調査

5,478 個の教材ファイルから 38 個の教材ファイル

Construction of a sample program database from teaching material files on the web

[†]Hideki Sugiura Graduate School of Engineering and Science, Shibaura Institute of Technology

[‡]Toru Sugimoto College of Engineering, Shibaura Institute of Technology

ルを抜粋した。抜粋した教材ファイルの内容を手作業で確認したところプログラムが書かれていると思われる箇所が 201 箇所存在した。

このそれぞれの箇所を抜き出して gcc でコンパイルしたところ、67 箇所エラーが発生し、オブジェクトファイルが作成されなかった。エラー原因を表 1 に示す。本研究ではエラーが発生した箇所を除いた 134 箇所をプログラムとみなして抽出の対象とする。

表 1. エラー原因の一覧

エラー原因	個数
文字化け	25 個
変数の宣言なし	6 個
脱字	4 個
ページを跨いだ記述	1 個
文字列の途中で改行	25 個
行番号の付加	6 個

3. 2. プログラム抽出のアルゴリズム

本研究では、C 言語の関数定義に用いられる中括弧{, }に着目してプログラムを抽出する。プログラム抽出のアルゴリズムを以下に記す。

- (1) 教材ファイルの先頭行を START 行とする。
- (2) START 行以降で最初に { が現れる行を BASE 行とする。もしそのような行がなければ終了。
- (3) BASE 行から前方に向かって 1 行ずつ、行頭が半角英数か #、* である限り遡り、プログラムの先頭に相当する FROM 行を求める。
- (4) BASE 行から後方へ 1 行ずつ { と } の出現回数の差が 0 より大きい限り読み進め、プログラムの末尾に相当する TO 行を求める。
- (5) TO 行の次の行を読み込み、{ が存在し、かつ定義される関数名の重複がなければこの行を BASE 行に再設定し、(4)へ戻る。
- (6) FROM 行から TO 行までを抜き出したファイルを作成し、gcc でコンパイルしてみる。エラーがなければ抽出し、エラーがあり FROM 行が (2) で設定した BASE 行よりも前にあるならば、FROM 行の次の行を FROM 行に再設定し (6) へ戻る。
- (7) TO 行の次の行を START 行に再設定し、(2) へ戻る。

3. 3. プログラム抽出の評価

抜粋した教材ファイル 38 個に対してプログラムの抽出を行ったところ、167 個のプログラムが得ら

れた。その内訳を表2に示す。表2よりプログラム抽出の適合率は74.9%、再現率は93.3%である。

表2. プログラムの抽出結果

		システム	
		抽出	抽出せず
人の判断	正	125	9
	誤	42	0

4. プログラムに関係する文章の抽出

4. 1. プログラムに関係する文章の調査

3.1. 節で述べた 134 個のプログラムのうち、76 個のプログラムの前後の文章を調査した。その結果、58 個のプログラムの前後にプログラムに関係する文章が記述されていた。その内訳は、そのプログラムで学べる学習項目を表す文が 55 文、プログラムの内容を説明する文が 60 文である。

またプログラムで学べる学習項目を表す文には、ほとんどの文の文末が名詞である (55 文中 54 文)、および文の単語数が 4 単語以上 10 単語以下である (55 文中 52 文) という特徴がみられた。

4. 2. 文章抽出の手法

プログラムに関係する文章を自動的に抽出するため、Support Vector Machine (SVM) を使用する。SVM は与えられた学習データを利用して、2 クラスのパターン識別器を構成する手法である。

4. 2. 1. 学習データの作成

学習データの作成には前節で述べた調査結果を用いた。抽出したプログラムの前後の約 200 文字から以下の 3 種類の文を用意した。

- (1) プログラムで学べる学習項目を表す文 (55 文)
- (2) プログラムの内容を説明する文 (60 文)
- (3) プログラムの前後の約 200 文字から (1) と (2) の文を除いた文 (1, 178 文)

(1) を正例データ、(2) と (3) を負例データとした学習データをプログラムの学習項目を表す文の学習データとした。また、(2) を正例データ、(1) と (3) を負例データとした学習データをプログラムの内容を説明する文の学習データとした。

4. 2. 2. 学習データの素性

SVM による分類のための素性として、それぞれの文に含まれる名詞と動詞を用いる。学習項目を表す文ではこれに加えて 4.1. 節で述べた特徴を用いて、単語数が 4 単語以上 10 単語以下であること、および文末が名詞であることを素性として用いる。

4. 2. 3. モデルデータの作成

前述の学習データおよび素性を SVM ツールの libsvm に与えて、プログラムの学習項目を表す文と内容を説明する文のモデルデータを作成した。カーネルは線形カーネルを使用し、それぞれの学習データを与えたときのコストは 0.25 とした。

4. 3. 文章抽出のアルゴリズム

作成した文章抽出のアルゴリズムを以下に記す。

- (1) プログラムの前後のそれぞれ 200 文字を取得する。
- (2) 文ごとに素性データを作成する。
- (3) libsvm とモデルデータを使用し、文ごとに素性データを 2 値分類する。
- (4) 2 値分類した結果からそれぞれの文を学習項目を表す文と内容を説明する文とそれ以外の文に分ける。

4. 4. プログラムに関係する文章抽出の評価

4.2. 節で述べた学習データを 5 つに分け、交差検定により評価した。その結果を表3と表4に示す。表3より、プログラムの学習項目を表す文の分類の正答率は 98.7%であり、この分類器を用いてプログラムの前後の約 200 文字に含まれる文の中からプログラムの学習項目を表す文を抽出する場合の適合率は 97.5%、再現率は 70.9%である。表4より、プログラムの内容を説明する文の分類の正答率は 96.8%であり、この分類器を用いてプログラムの前後の約 200 文字に含まれる文の中からプログラムの内容を説明する文を抽出する場合の適合率は 68.8%、再現率は 55.0%である。

表3. 学習項目を表す文の交差検定の結果

		SVM	
		該当	非該当
人の判断	正	39	16
	誤	1	1237

表4. 内容を説明する文の交差検定の結果

		SVM	
		該当	非該当
人の判断	正	33	27
	誤	15	1218

5. まとめ

本研究では Web 上の教材ファイルからプログラムとその前後に書かれたプログラムに関係する文章を抽出する方法の提案を行った。

プログラム抽出アルゴリズムで 9 個のプログラムが抽出できなかった。隣接したプログラムの境界が正しく認識できなかったことが原因と考えられる。また適合率が 7 割に留まった原因としては、配列の初期化や構造体のみのプログラムをコンパイルしてもエラーが起きないため誤って抽出されるためと考えられる。

プログラムに関係する文章抽出の再現率が悪かった原因として学習データのそれぞれの正例に対して負例が約 20 倍と多いため、学習が負例に引きずられることが考えられる。この問題を解決するためにオーバーサンプリング、またはアンダーサンプリングを行う必要があると考えられる。

参考文献

- [1] 中森慎平, 杉本徹: “学習項目オントロジーに基づく情報工学教材の体系化”, 情報処理学会第74回全国大会, 2012-03