プログラミング教育支援に向けた サンプルプログラム推奨システム

東京情報大学　総合情報学部
宇田川 佳久

目　次

1. 研究の背景と関連研究

- サンプルプログラムは新技術の習得に有効である。
- サンプルコードが膨大であることから目視での確認は非効率的。
- サンプルプログラムを推奨するシステムの研究開発が盛んである。
- ソフトウェア開発における推奨システムをRSSE (Recommendation System for Software Engineering)と呼ぶ（文献1）。

参考文献

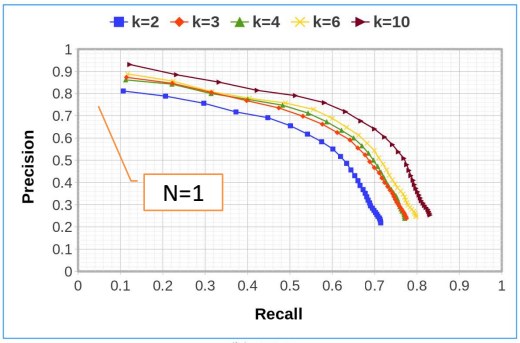
1. Robillard, M., Walker, R., Zimmermann, T., "Recommendation systems for software engineering," IEEE Software, 8, pp. 80-86, Jul. 2010.
2. Gasparic , N. and Janes A. , "What Recommendation Systems for Software Engineering Recommend: A Systematic Literature Review," Journal of Systems and Software 113, Nov. 2015, DOI:10.1016/ j.jss.2015.11.036
3. Diamantopoulos,T. and Symeonidis L. A., "Mining Source Code for Component Reuse," Mining Software Engineering Data for Software Reuse, pp 133–174, Mar. 2020.
4. Katirtzis, N., Diamantopoulos, T., and Sutton, C., "Summarizing Software API Usage Examples using Clustering Techniques," *Proc. of the 21st International Conference on Fundamental Approaches to Software Engineering*. vol. 10802, Springer, Cham, Thessaloniki, Greece, pp. 189-206, Apr. 2018, DOI: 10.1007/978-3-319-89363-1_11
5. Phuong T. N., Juri Di R., Claudio Di S., Davide Di R., and Massimiliano Di P., "Recommending API Function Calls and Code Snippets to Support Software Development," IEEE Transactions on Software Engineering, Vol. 48, pp. 2417 - 2438, July 2022. DOI: 10.1109/TSE.2021.3059907
6. Hsu, S.-K. and Lin, S.-J., "Mining Source Codes to Guide Software Development," Asian Conference on Intelligent Information and Database Systems (ACIIDS 2010), pp 445–454.

関連研究の概要

① 文献2：　RSSEに関する46本の研究開発論文をサーベイした。ソースコードを対象とした論文が21本で最多。
② 文献3：　インターネットのソフトウェアリポジトリを対象とし、コード断片を提示することで，類似するサンプルコードを推奨する。類似度の計算は、ベクトル空間と Levenshtein距離に基づいて検索する.
③ 文献4：　API呼び出しシーケンスを抽出した後に，クラスタ化し，APIの利用サマリ（snippet）を作成するアルゴリズムについて論じた。類似度の計算は、最長共通部分列（LCS）を使用している。
④ 文献5：　コーディングの工程でAPIの利用法を適時に提示するFOCUS の提案と評価実験結果を示した。類似度の計算は、tf-idfを使用している。

文献5より抜粋

k: 類似するAPの数
N: 推奨リストの表示数



(b) C1.2

Phuong T. N., Juri Di R., Claudio Di S., Davide Di R., and Massimiliano Di P., "Recommending API Function Calls and Code Snippets to Support Software Development,"

従来の研究との違い

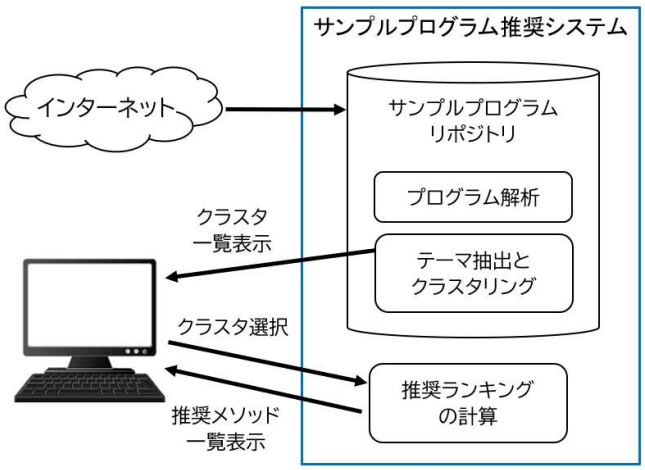
(1) 本システムでは，極大頻出アイテム集合に基づくクラスタリングにより，自動的にテーマを抽出し，テーマごとにサンプルプログラムをクラスタリングして利用者に推奨できる．

(2) Tf-idfに基づく重み付けにより，レアな API を含み，かつ，APIの個数が多いサンプルプログラムが上位にランクされる．

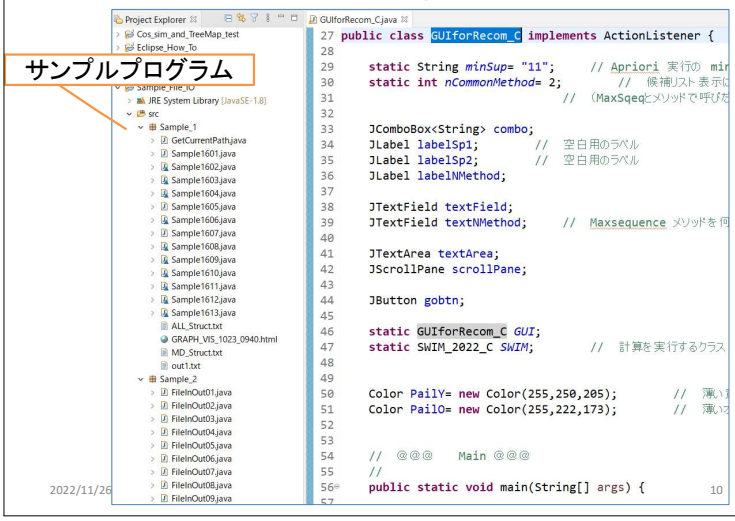
(3) クラスタリングに関与する API に大きな重みを付けるようにtf-idfの計算結果を調整することで，クラスタリングのテーマに合致したサンプルプログラムが上位にランクされる．

目　次

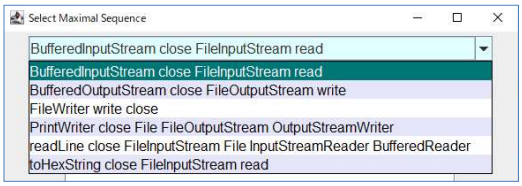
2. 推奨システムの概要



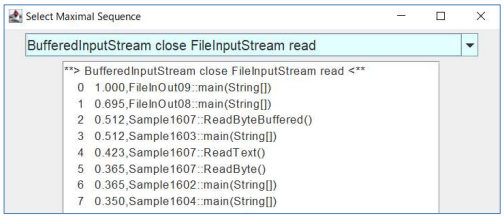
推奨システムは Eclipse で稼働する



1. リポジトリを指定して実行すると、クラスタの一覧が表示される。



2. クラスタを指定すると、サンプルが推奨順に表示される。



目　次

3.1 構造情報の抽出

- Javaプログラムの構文解析では，JDT（Java Development Tools）-Core の Scanner クラスを使用した．

```
public class LongestCommonSubsequence_2 {

    int lcs( char[] X, char[] Y, int m, int n ) {
      int L[][] = new int[m+1][n+1];

      for (int i=0; i<=m; i++) {
        for (int j=0; j<=n; j++) {
          if (i == 0 || j == 0)
              L[i][j] = 0;
          else if (X[i-1] == Y[j-1])
              L[i][j] = L[i-1][j-1] + 1;
          else
              L[i][j] = max(L[i-1][j], L[i][j-1]);
        }
      }
    return L[m][n];
    }

    int max(int a, int b) {
      return (a > b)? a : b;
    }
}
```

1	1 .	17	36	new
2	2 ++	18	40	static
3	4 +	19	42	1
4	5 -	20	47	
5	6 [	21	49	{
6	12 <=	22	57	public
7	15 >	23	63	:
8	19 ==	24	67	]
9	22 n	25	68	class
10	23 (	26	73	=
11	25 ;	27	82	for
12	26)	28	83	if
13	29 ?	29	84	return
14	31 ||	30	105	char
15	32	31	109	int
16	33 }	32	112	void
		33	114	else

この研究で開発した構文解析機能の概要

- 一行に複数のメソッドがある場合にも対応している。
- println() や printStackTrace() など，共通的に使用されるメソッドは，除外する機能を作り込んだ．

```
private static void timeMeasure(String MID) {
  try {
    System.out.println(MID);
    Method method = Class.forName("Sample1607").getMethod(MID);
    long startTime = System.currentTimeMillis();
    method.invoke(null); // メソッドの実行。
    long endTime = System.currentTimeMillis();
    System.out.println(MID+" ReadTime : "+(endTime - startTime)+" msec");
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

メソッド抽出処理の概要（事例）

```
1  package Sample_1;
2  import java.io.*;
3  import java.lang.reflect.Method;
4
5  class Sample1607 {
6      static String FName = "c:\\temp\\Apriori\\MD_Struct.txt";
7      public static void main(String args[]) {
8          timeMeasure("ReadByte");
9          timeMeasure("ReadByteBuffered");
10         timeMeasure("ReadText");
11         timeMeasure("ReadTextBuffered");
12     }
13     public static void ReadByte() throws IOException {
14         FileInputStream fis = new FileInputStream(FName);
15         int code;
16         while ((code = fis.read()) != -1) {
17             Integer.toHexString(code);
18         }
19         fis.close();
20     }
           中略
39     public static void ReadTextBuffered() throws IOException {
40         BufferedReader br = new BufferedReader(new FileReader(FName));
41         int data;
42         while ((data = br.read()) != -1) {
43             Character.toChars(data);
44         }
45         br.close();
46     }
47     private static void timeMeasure(String MID) {
48         try {
49             System.out.println(MID);
50             Method method = Class.forName("Sample1607").getMethod(MID);
51             long startTime = System.currentTimeMillis();
52             method.invoke(null); // メソッドの実行。
53             long endTime = System.currentTimeMillis();
54             System.out.println(MID+" ReadTime : "+(endTime - startTime)+'
55         } catch (Exception e) {
56             e.printStackTrace();
```

```
Sample1607::main(String[])
  timeMeasure,4
Sample1607::ReadByte()
  FileInputStream,1
  close,1
  read,1
  toHexString,1
Sample1607::ReadByteBuffered
  BufferedInputStream,1
  FileInputStream,1
  close,1
  read,1
  toHexString,1
Sample1607::ReadText()
  FileInputStream,1
  InputStreamReader,1
  close,1
  read,1
  toChars,1
Sample1607::ReadTextBuffered
  BufferedReader,1
  FileReader,1
  close,1
  read,1
  toChars,1
Sample1607::timeMeasure(Stri
  currentTimeMillis,2
  forName,1
  getMethod,1
  invoke,1
```


メソッド抽出処理結果

クラスタリング用

```
timeMeasure
FileInputStream close read toHexString
        中略
BufferedReader FileReader close read toChars
currentTimeMillis forName getMethod invoke
```

類似度の計算用

```
Sample1607::main(String[])
  timeMeasure,4
Sample1607::ReadByte()
  FileInputStream,1
  close,1
  read,1
  toHexString,1
    中略
```

```
Sample1607::ReadTextBuffered()
  BufferedReader,1
  FileReader,1
  close,1
  read,1
  toChars,1
Sample1607::timeMeasure(String)
  currentTimeMillis,2
  forName,1
  getMethod,1
  invoke,1
```


3.2 Aprioriアルゴリズムと極大頻出集合

- データベースD： $D=\{t_1, t_2, \ldots, t_n\}$
- アイテムの集合 t_i： $\{I_1, \ldots, I_m\}$
- Support(X)でデータベースDに出現するXの頻度（出現個数）：
  $Support(X)= |\{ t_i \in D : X \subseteq t_i \ \&\ 1 \le i \le n \}|$

- Apriori アルゴリズムは、Support(X)が指定された最小サポート数（minSup）以上の頻度で発生する“頻出アイテム集合”を効率よく列挙する。

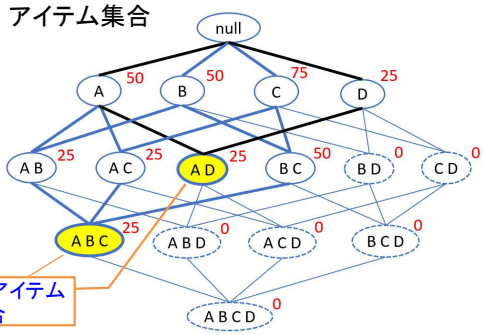
頻繁に発生する要素集合（パターン）を抽出する。

- 頻出アイテム集合は多くのアイテムから構成されていることから，極大頻出アイテム集合が考案された。
- 【定義】極大頻出アイテム集合とは，X にどんなアイテム i を1つ加えたアイテム集合Y = X ∪{ i } も頻出でなくなるような頻出アイテム集合である．

アイテム集合（データベース）

```
A B C
A D
C
B C
```

最小支持度minSup を25%で抽出したアイテム集合



3.3 極大頻出集合を使ったクラスタリング機能の実装

fpgrowth.exe（文献9）を本システム（Java）から呼び出すことで実装した.

生成された極大頻出集合の例

```
1 FileWriter write close (11.7647)
2 toHexString close FileInputStream read (11.7647)
3 BufferedOutputStream close FileOutputStream write (11.7647)
4 PrintWriter close File FileOutputStream OutputStreamWriter (11.7647)
5 BufferedInputStream close FileInputStream read (11.7647)
6 readLine close File FileInputStream InputStreamReader BufferedReader (14.7059)
```


- 極大頻出集合（クラスタ）に対応するメソッド名を編集する。
- 以下は、2個以上の要素を含むサンプルコードの一覧。

```
0  BufferedInputStream close FileInputStream read
     FileInOut08::main(String[])
     FileInOut09::main(String[])
     Sample1602::main(String[])
     Sample1603::main(String[])
     Sample1604::main(String[])
     Sample1607::ReadByte()
     Sample1607::ReadByteBuffered()
     Sample1607::ReadText()
1  BufferedOutputStream close FileOutputStream write
     FileInOut07::main(String[])
     FileInOut09::main(String[])
     Sample1608::main(String[])
     Sample1609::main(String[])
     Sample1610::main(String[])
     Sample1613::byteBufferedWrite()
     Sample1613::byteWrite()
```


目　次

4. 推奨ランキングの計算

- tf-idf法に基づいて，プログラムの推奨順位を計算する。
- 類似度を、Tf は「単語出現頻度」とIdf は「逆文書頻度」の積で算出する。
  - Tf=　単語iの文書dにおける出現回数 / 文書dにおける全単語の出現回数の和
  - Idf(w)= log（　総文書数 / ある単語wを含む文書の数）

Idfの計算例

メソッド名	出現回数	idf
BufferedInputStream	4	0.574
BufferedOutputStream	1	1.176
BufferedReader	6	0.398
File	6	0.398
FileInputStream	12	0.097
FileOutputStream	2	0.875
FileReader	3	0.699
InputStreamReader	6	0.398
OutputStreamWriter	1	1.176
PrintWriter	1	1.176
close	15	0
flush	1	1.176
format	1	1.176
length	1	1.176
read	11	0.135
readLine	4	0.574
toChars	2	0.875
toHexString	4	0.574
write	1	1.176

クラスタリングに関与するメソッドのidf値を調整する。

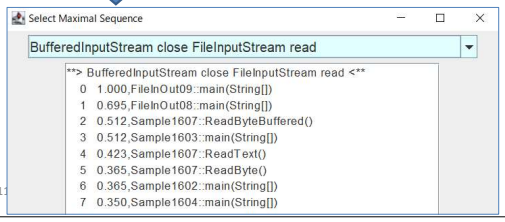
API名	調整前	調整後
BufferedInputStream	0.574	1.75
FileInputStream	0.097	1.273
close	0	1.176
read	0.135	1.311

推奨値の計算

推奨値= 調整後idf値 * tf

id	推奨値	サンプルプログラム名
0	8.26	FileInOut08::main(String[])
1	11.89	FileInOut09::main(String[])
2	4.334	Sample1602::main(String[])
3	6.084	Sample1603::main(String[])
4	4.158	Sample1604::main(String[])
5	4.334	Sample1607::ReadByte()
6	6.084	Sample1607::ReadByteBuffered()
7	5.033	Sample1607::ReadText()

推奨値の最大を 1になるように計算する。



目　次

5. 実験結果

推奨値 1.000

```
public class FileInOut09 {
    public static void main(String[] args) {
        String inputFileName = "input.dat";
        String outputFileName = "output.dat";
        File inputFile = new File(inputFileName);
        File outputFile = new File(outputFileName);
        try {
            FileInputStream fis = new FileInputStream(inputFile);
            BufferedInputStream bis = new BufferedInputStream(fis);
            FileOutputStream fos = new FileOutputStream(outputFile);
            BufferedOutputStream bos = new BufferedOutputStream(fos);
            byte[] buf = new byte[1024];
            int len = 0;
            while ( ( len = bis.read(buf) ) != -1 ) {
                bos.write(buf, 0, len);
            }
            bos.flush();    bos.close();    bis.close();
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

推奨値 0.350

```
class Sample1604 {
    public static void main(String[] args) throws IOException {
        String inputFile = "c:¥¥dev¥¥java¥¥Sample1601.txt";
        InputStreamReader isr = new InputStreamReader(
                    new FileInputStream(inputFile), "SJIS");
        int data;
        while ((data = isr.read()) != -1) {
            System.out.print((char)data);
        }
        isr.close();
    }
}
```


6. おわりに

1. 呼び出しているAPIに基づいて
   ① サンプルプログラムのクラスタリングを行い、
   ② クラスタリングに関連するサンプルプログラムの推奨、
   を行うシステムについて述べた。
2. クラスタリングには、Aprioriアルゴリズムの極大頻出集合を使用し、ソフト・クラスタリングを実現している。
3. 推奨値の計算は，tf–idf に独自の重み付けを行っている。
4. 今後は、色々な分野のサンプルプログラムを使った実験を行う。
5. また、教育現場への適用を計画している。

ご清聴ありがとうございました。