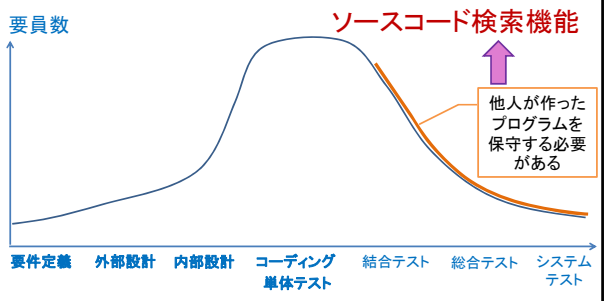
シーケンスに基づく検索モデルの検索精度について

東京工芸大学工学部コンピュータ応用学科
宇田川　佳久

目　次

1. 研究の背景と概要 (1/3)

情報システム開発のイメージ



1. 研究の背景と概要 (2/3)

実務面での応用

①バグあるいは脆弱なコードを探す
（品質の高いシステムを開発する）

②プログラム理解を支援する
（第3者が書いたコードを保守する）

③特定のメソッドを使ったコードを探す
（テクニックの向上）

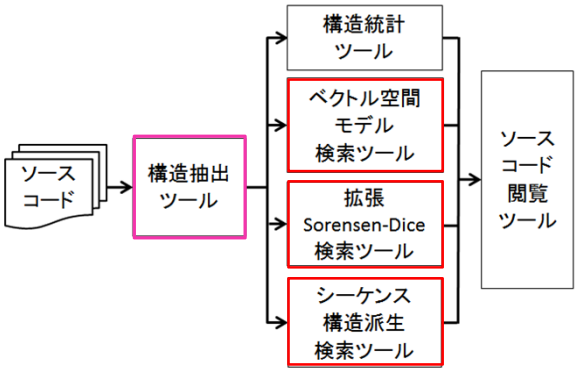
1. 研究の背景と概要 (3/3)

ソースコードの類似検索手法

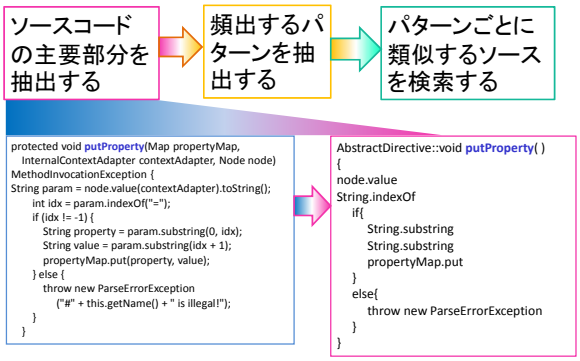
① テキストに基づく検索 –変数名の違いに影響を受ける
② トークンに基づく検索
–変数名の違いは除去できるが、局所的な検索の留まる
③ 構造に基づく検索 ― 本研究のアプローチ
–文法に沿った検索ができるが、処理が複雑

目　次

開発したツールと実験対象の概要



開発したツールと実験対象の概要



実験対象の主なメトリックス

現時点で Struts 2 と Tomcat 7 のソースコードで検証中

メトリックス	Struts 2	Tomcat 7
ファイル数	368個	1,100個
クラス数	414個	1,681個
メソッド数	2,677個	15,640個
宣言文と実行文の行数	21,543行	177,724行
コメント行数	17,954行	108,167行
ソース総行数	46,100行	334,457行
最大ネスト数:	8	10
最大循環的複雑度	55	133

目　次

構造抽出ツールの主な機能

対象は Java言語　— Java言語固有の機能を検索に反映している

(1) クラス名，メソッド名と引数　— オブジェクト型

① クラス名::メソッド名(引数)
② クラス名:匿名クラス名:メソッド名(引数)

```
interface IntF{
  void printHello(int x);
}
class HelloAno {
  public static void main(String[] args) {
    IntF object1 = new IntF() {
      public void printHello(int x) {
        System.out.println("匿名クラス= "+ x);
      };
    object1.printHello(767);
  }
}
```

匿名クラスの例

構造抽出ツールの主な機能

(2) Javaの制御文

1. if文（else, else if）
2. switch文
3. while文
4. do while文
5. for文
6. break文
7. continue文
8. return文
9. throw文
10. synchronized文
11. try文（catch，finally）

構造抽出ツールの主な機能

(3) メソッドの中で呼び出しているメソッド名

変数名.メソッド名
⇒ クラス名.メソッド名
または
データ型.メソッド名

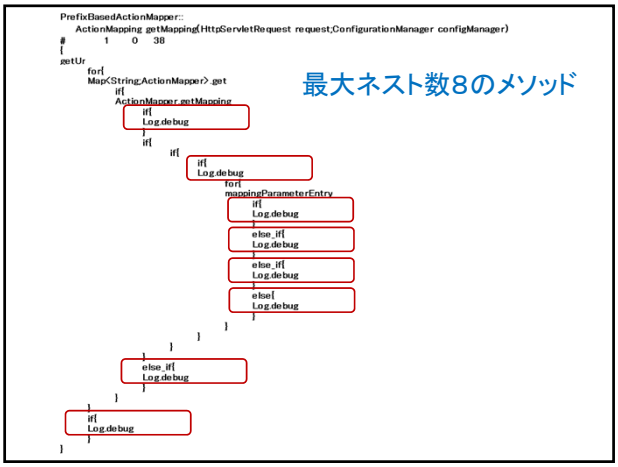
別物として扱う

変数名1.add (<String>) ⇒ **LinkedList <String>.**add(<String>)

変数名2.add (<String>) ⇒ **List<String>.**Add (<String>)

制御文とメソッド名の構造の抽出結果

```
Settings::static Settings getDefaultInstance()
#   8  3 #
{
   if{
   DefaultSettings
      try{
      get
         if{
            try{
            (Settings) ObjectFactory.getObjectFactory()
            }
            catch{
            Logger.error
            }
         }
      }
      catch{
      }
   }
}
```



最大ネスト数8のメソッド

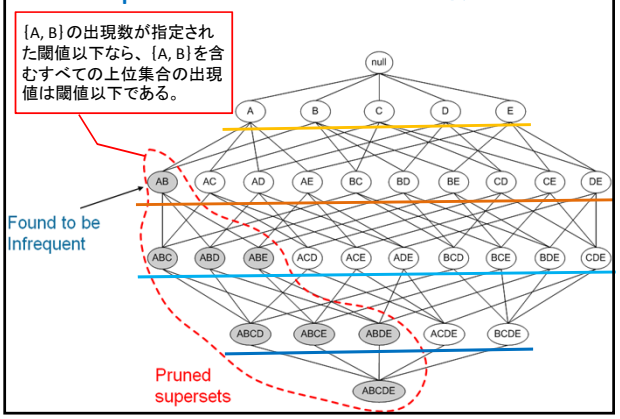
目　次

1. 研究の背景と概要
2. 開発したツールと実験対象の概要
3. ソースコードの構造抽出
4. 頻出シーケンスの抽出
5. ソースコードの類似指標
6. 検索実験と評価
7. まとめと今後の研究方針

頻出シーケンスの抽出

- 目的
  適切な検索条件を見つけるために、プログラムに数多く含まれる構造を抽出する
- 方法
  - Aprioriアルゴリズム（Agrawal 1994）は、集合内の頻出アイテム集合（Frequent Itemsets）を検出する
  - 頻出シーケンス集合を検出するように改良した

Aprioriアルゴリズムの概念



頻出シーケンスの抽出

```
List <statement[]>  MS, LS, Tk , Ck;
LS= Φ;
   k=1;
   Tk= { i | i ∈ {Control Statements} };
  repeat
      k=k+1
      Ck= Retrieve_Candidates( MS, Tk );   // A set of sequences that begins Tk
      Tk= Φ;
     for ( each c∈Ck ){
        if ( at least one element of c is not a control statement  &&
             |c| ≧ minSup ){
           LS.add(c);  Tk.add(c)  };
     }
  until Tk= Φ;
Result= LS;
```

最初の要素は制御文

この処理は、制御文だけから構成されるシーケンスを除外している。
アプリケーション依存の処理。

抽出した頻出シーケンス
対象ソースは Struts 2

No	シーケンス	検出数
1	if{ → }	162
2	if{ → addParameter	102
3	if{ → addParameter → }	96
4	if{ → addParameter → } → if{	67
5	if{ → if{	59
6	if{ → } → }	47
7	if{ → Logger.debug	42
8	if{ → Logger.warn	41
9	if{ → Logger.warn → }	41
10	if{ → Logger.debug → }	36
11	if{ → Logger.warn → } → }	35
12	throw operationNotSupported → }	33
13	if{ → } → if{	32
14	else{ → }	27
15	if{ → try{	26
16	for{ → if{	23
17	if{ → } → else{	23
18	if{ → } → } → }	23
19	catch{ → Logger.error	20
20	else{ → } → }	20
21	catch{ → if{	19
22	if{ → for{	18
23	else_if{ → }	17
24	catch{ → Logger.error → }	17
25	if{ → } → if{ → }	15
26	if{ → StringBuilder.append	14
27	if{ → addParameter → } → }	14
28	if{ → String.substring	13
29	catch{ → if{ → Logger.warn	13
30	catch{ → if{ → Logger.warn → }	13
31	if{ → findValue	11
32	for{ → iterator.next	10
33	for{ → if{ → }	10
34	for{ → if{ → } → }	10
35	if{ → StringBuilder.append → }	9
36	else_if{ → addParameter → }	8
37	if{ → findValue → }	7
38	while{ → StringTokenizer.nextToken	6
39	if{ → addParameter → } → else{	6
40	while{ → Enumeration.nextElement	5
41	if{ → findString	4
42	for{ → List<String>.add	4
43	while{ → Iterator.next	4
44	if{ → findString → }	4
45	for{ → StringBuilder.append	3

目　次

類似検索機能の比較方針

- ソースコードは，プログラミング言語としての文法に準拠した文書
- 自然言語で記述された一般的な文書との違いを比較する

一般的な文書⇒　ベクトル空間モデル
Sørensen-Dice（Jaccard）係数

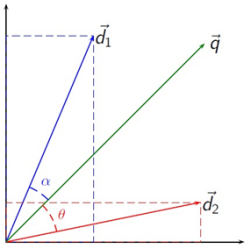
ソースコード　⇒　構造派生検索モデル

ベクトル空間モデルの概要

① 文書と検索条件を単語のベクトルとして表現する

$$d_j = (w_{1,j}, w_{2,j}, \ldots, w_{N,j})$$

$$q = (w_{1,q}, w_{2,q}, \ldots, w_{N,q})$$



② 文書と検索条件の類似度を2つのベクトルの Cos で計算する

$$\text{Similarity}(\boldsymbol{d_j}, \boldsymbol{q}) = \cos(\boldsymbol{d_j}, \boldsymbol{q}) = \frac{\sum_{i=1}^{N} w_{i,j} * w_{i,q}}{\sqrt{\sum_{i=1}^{N} w_{i,j}^2} * \sqrt{\sum_{i=1}^{N} w_{i,q}^2}}$$

Sørensen-Dice（Jaccard）係数

$$S_{\text{Sorensen-Dice}} = 2a / (2a + b + c)$$

a = number of elements common to both A and B
b = number of elements unique to A
c = number of elements unique to B

$$S_{\text{Sorensen-Dice}}(a, b) = \frac{2|a \cap b|}{2|a \cap b| + |a \cap \neg b| + |\neg a \cap b|}$$

N 個の集合に関する Sørensen-Dice 類似度

$$S_{\text{Sorensen-Dice}}(X_1, X_2, \ldots, X_n) = \frac{n|X_1 \cap X_2 \cap \ldots \cap X_n|}{\sum_{r=0}^{r=n-1} |(n-r)\,SetComb(X_1 \cap X_2 \cap \ldots \cap X_n,\ r)|}$$

n個から r 個取り出す組合わせ

シーケンス構造派生検索モデルの類似度の定義

- N個の集合に対するSørensen-Dice類似度を、N個の要素のシーケンス(→)に置き換える

$$S_{DSRM}\left([S_1 \rightarrow S_2 \rightarrow \cdots \rightarrow S_m], [T_1 \rightarrow T_2 \rightarrow \cdots \rightarrow T_n]\right) = \frac{n \,||\, [S_1 \rightarrow S_2 \rightarrow \cdots \rightarrow S_m],\ [T_1 \rightarrow T_2 \rightarrow \cdots \rightarrow T_n] \,||}{\sum_{r=0}^{r=n-1} (n-r) \,||\, [S_1 \rightarrow S_2 \rightarrow \cdots \rightarrow S_m],\ SqcComb([T_1 \rightarrow T_2 \rightarrow \cdots \rightarrow T_n],\ r) \,||}$$

n個要素のシーケンスを含む数

n - r 個要素のシーケンスを含む数。 カウント後、該当するシーケンスを除外する。

n個から r 個取り出したシーケンス

目　次

1. 研究の背景と概要
2. 開発したツールと実験対象の概要
3. ソースコードの構造抽出
4. 頻出シーケンスの抽出
5. ソースコードの類似指標
6. 検索実験と評価
7. まとめと今後の研究方針

検索実験と評価

	Struts 2	Tomcat 7
頻出シーケンスの抽出	実施済み	実施中
ソースコードの類似検索	実施済み	実施中

検索実験と評価 (Struts 2)

検索条件：

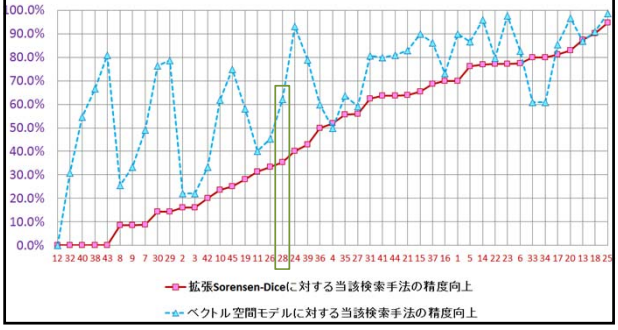
ベクトル空間モデル、Sorensen-Diceモデル： { *If{ , String.substring* }

構造派生モデル； [*if{ → String.substring*]

No	メソッド名	ベクトル空間モデル	拡張Sorensen-Diceモデル			シーケンス構造派生検索モデル(DSRM)			ソース行数
		類似度	類似度	完全一致	部分一致	類似度	完全一致	部分一致	
1	MultiselectInterceptor::String intercept()	0.280	0.667	2	1	0.667	2	1	17
2	RegexPatternMatcher::RegexPatternMatche	0.204	0.667	2	1	0.667	2	1	23
3	AbstractDirective::void putProperty()	0.595	0.667	2	1	0.667	2	1	11
4	ServletUrlRenderer::String extractQueryStri	0.439	0.500	2	2	0.500	2	2	13
5	DefaultActionMapper::void handleSpecialPa	0.312	0.500	2	2	0.500	2	2	21
6	CheckboxInterceptor::String intercept()	0.329	0.400	2	3	0.400	2	3	24
7	RequestUtils::static String getServletPath()	0.673	0.667	4	2	0.333	2	4	19
8	Restful2ActionMapper::ActionMapping getS	0.340	0.300	6	14	0.300	6	14	80
9	ServletUrlRenderer::void renderUrl()	0.190	0.286	2	5	0.286	2	5	48
10	ServletUrlRenderer::void renderFormUrl()	0.210	0.500	6	6	0.167	2	10	66
11	DefaultActionMapper::String getUriFromAc	0.358	0.286	4	10	0.143	2	12	43
12	ResourceUtil::static String getResourceBas	0.662	1.000	2	0	0	0	2	7
13	TagUtils::static String buildNamespace()	0.300	0.667	2	1	0	0	3	16
14	PrepareOperations::String getUri()	0.474	0.667	2	1	0	0	3	12
15	DefaultActionMapper::String getUri()	0.435	0.667	2	1	0	0	3	13
16	RestfulActionMapper::ActionMapping getM	0.304	0.333	2	4	0	0	6	36
17	Include::static String getContextRelativePat	0.299	0.333	2	4	0	0	6	35
18	Component::String getComponentName()	0.347	0	0	1	0	0	1	6
19	StrutsConversionErrorInterceptor::boolean	0.307	0	0	6	0	0	6	22
20	Component::String completeExpressionIfAl	0.269	0	0	1	0	0	1	6
21	ApplicationMap::boolean equals()	0.269	0	0	1	0	0	1	7

構造派生モデルの効果 (Struts 2)

- 拡張Sorensen-Dice係数に対して平均46.4% 、最大94.6%
- ベクトル空間モデルに対して平均66.8% 、最大98.7%



目　次

1. 研究の背景と概要
2. 開発したツールと実験対象の概要
3. ソースコードの構造抽出
4. 頻出シーケンスの抽出
5. ソースコードの類似指標
6. 検索実験と評価
7. まとめと今後の研究方針

まとめと今後の研究方針

1. ソースコード抽出処理の改良
2. 頻出シーケンス抽出アルゴリズムの改良
3. 検索用操作画面の開発
4. 実験対象の拡大
   - ① 他のJavaオープンソースへの適用
   - ② Java以外の言語への対応

ご清聴ありがとうございました