

T-複雑度によるソースコード中の類似文字列の検出評価

高橋 圭一[†] 戒田 高康[†]

Mark Titchener によって提案された T-符号に基づく T-複雑度をソースコード中に含まれる類似文字列の検出に適用を試みる。具体的には、平均 7 行程度の Java コード片に対して圧縮ツールの 1 つである bzip2 を用いて圧縮率を求め、ソースコード中に類似文字列が含まれる傾向を調べる。この値と T-複雑度との相関を求めることにより、T-複雑度による類似文字列の検出の可能性を評価する。実験結果より、bzip2 の圧縮率と T-複雑度および T-情報量には負の相関があり、bzip2 と T-平均情報量の相関係数は 0.64 と近い傾向があることがわかった。

1. はじめに

1984 年に Mark Titchener によって提案された自己同期する可変長符号である T-符号がある[1]。この T-符号をもとに系列の複雑さをあらわす指標の 1 つである T-複雑度などが提案された[2]。T-分解アルゴリズムと呼ばれる変換によって系列を部分系列に分解でき、この部分系列をもとに系列の複雑さをあらわす T-複雑度が得られる。T-複雑度を利用して乱数検定、情報圧縮、あるいは生体信号やネットワークデータを用いたイベント検知などの応用が研究されているが、我々が調査した限りソースコードに適用した研究はない。本稿では、ソースコード中の類似文字列の検出の可能性について、OSS から抽出した Java コード片を対象として実験により評価した結果を報告する。

2. T-複雑度・T-情報量・T-平均情報量

2.1 定義[3]

有限個からなるアルファベット A により構成されるアルファベット列 A^+ を考える。ここで、 $|A|$ を集合 A の要素数として、 $r = |A| \geq 2$ とする。接頭符号 $S \subset A^+$ に対して、その符号語 $p \in S$ および $k \in \mathbb{N}^+$ によって決まる S の T-増大

(T-augmentation) された (接頭) 符号 $S_{(p)}^{(k)}$ を

$$S_{(p)}^{(k)} = \bigcup_{i=0}^k p^i S \setminus \bigcup_{i=0}^k p^i \quad (1)$$

で定義する。ここで、 p^i は符号語 p を i 回繰り返して接続したアルファベット列で (特に、 p^0 は空のアルファベット列をあらわす)、 $p^i S$ はアルファベット列 p^i と S の各符号語とをそれぞれ接続して生成される語 (アルファベット列) 全体の集合を意味する。また、 $A \setminus B$ は集合 A の要素から集合 B の要素を除いた集合を意味する。

初期の接頭符号を形式的に $S_0^0 = A$ より開始して、逐次的に T-増大することにより、 $q \geq 1$ として

$$S_{(p_1, p_2, \dots, p_q)}^{(k_1, k_2, \dots, k_q)} = \left(S_{(p_1, p_2, \dots, p_{q-1})}^{(k_1, k_2, \dots, k_{q-1})} \right)_{(p_q)}^{(k_q)} \quad (2)$$

の表現において、 $p_1 \in A$, $p_i \in S_{(p_1, \dots, p_{i-1})}^{(k_1, \dots, k_{i-1})}$ ($i = 2, 3, \dots, q$) とする。このように定義される符号 (列) を T-符号 (列) と呼び、

$$S_{(p_1, \dots, p_q)}^{(k_1, \dots, k_q)} = \bigcup_{(i_1, \dots, i_q) = (0, \dots, 0)}^{(k_1, \dots, k_q)} p^{i_q} \dots p^{i_1} S \setminus \bigcup_{(i_1, \dots, i_q) = (0, \dots, 0)}^{(k_1, \dots, k_q)} p^{i_q} \dots p^{i_1} \quad (3)$$

でも表現できる。

アルファベット A 上の T-符号 S が $p = (p_1, \dots, p_q)$ と $k = (k_1, \dots, k_q)$ により構成されたとき、T-符号 $S = S_p^k$ の T-手順と呼ぶ。

T-符号 S が $S = S_p^k$ の場合、その符号 S の T-複雑度 $C_T(S)$ を

$$C_T(S) = \log_2 \left(\prod_{i=1}^q (k_i + 1) \right) = \sum_{i=1}^q \log_2 (k_i + 1) \quad (4)$$

で定義する。

任意の T-符号 $S = S_p^k$ に対して、その k の長さ q の最小性を仮定した場合、その T-手順の一意性は確認されており、標準 T-手順と呼ぶ。したがって、与えられた T-符号に対する T-複雑度は、その標準 T-手順の T-複雑度により定義することで一意に決まる。なお、T-複雑度 $C_T(S)$ の単位は *taugs* (T-augmentation steps) である。

ここで、アルファベットの列 $x \in A^+$ の T-情報量 $I_T(x)$ をその

T-複雑度 $C_T(x)$ と $\text{li}(n) = \int_0^n \frac{du}{\ln u}$ の逆関数 $\text{li}^{-1}(x)$ を用いて

$$I_T(x) = \text{li}^{-1}(C_T(x)) \quad (5)$$

で定義する。また、 $|x|$ をアルファベット列 x の長さとする、T-平均情報量 $H_T(x)$ を

$$H_T(x) = \frac{I_T(x)}{|x|} = \frac{\text{li}^{-1}(C_T(x))}{|x|} \quad (6)$$

により定義する。

2.2 計測ツール

本稿では、[4]より取得した計測ツールである *tcalc* を用いて T-複雑度、T-情報量、T-平均情報量を求める。*tcalc* の T-分解アルゴリズムでは与えた文字列をバイト単位で処理する。そのため、ソースコードに含まれる英数字や演算子等の記号のほか、空白、タブ、改行も文字として扱う。し

[†] 近畿大学産業理工学部情報科学
Department of Information and Computer Sciences, Faculty of
Humanity-Oriented Science and Engineering, Kinki University

たがって、複数行にわたるソースコードも連続した1つの文字列として扱う。1バイトの文字を対象とするため、tcalcに与えるアルファベットの要素数は256である。

3. 実験

3.1 方法

ソースコード中に「類似文字列が含まれる」ことを「規則性を有した文字並びが含まれる」と換言できると考える。そこで類似文字列の有無を圧縮ツールを用いた圧縮率(=圧縮後ファイルサイズ / 圧縮前ファイルサイズ)により計測する[5]。つまり圧縮率が高いほどソースコード中に類似文字列が多く含まれていると考える。続いてこの圧縮率とT-複雑度等との相関を求める。圧縮ツールとしては bzip2, compress, gzip の3つを用いる。また、[5]を参考にエントロピーも求めて圧縮率との相関を調べる。

3.2 手順

Buse氏が公開しているOSSから抽出した100個のJavaコード片(平均7行程度)[6]に対して、tcalcを用いてT-複雑度、T-情報量、T-平均情報量を求める。さらにコード片について bzip2, compress, gzip を用いて圧縮率を求める。bzip2による圧縮率とその他の値についてスピアマンの順位相関係数を求める。

3.3 結果

結果を表1に示す。compressおよびgzipによる圧縮率は高い相関を示している。圧縮方式は異なっても圧縮しやすさには同様の傾向があると考えられる。T-複雑度およびT-情報量はともに bzip2 と負の相関がある。T-平均情報量は正の相関でありT-複雑度より大きい値を示している。エントロピーは相関がみられず[5]と同様の結果が確認できた。T-平均情報量の低いコード片:T-平均情報量が最も低い3つのコード片を順に示す(図1~3)。なお、順番は異なるものの、これらは bzip2 による圧縮率が最も高い3つのコード片でもある。いずれのコード片も類似文字列を多く含んでいることがわかる。

T-複雑度が負相関となる要因:文字並びに規則性があれば、圧縮ツールと同様にT-複雑度は低下する傾向があるが、T-複雑度は文字列長に影響を受けることもわかっている(T-複雑度とコード片長との相関係数は0.89)。文字並びの規則性より文字列長による影響を強く受けたため負の相関になったと考える。図4にT-複雑度が最も低いコード片を示す。コード片1~3のような文字並びに規則性がみられないものの、コード片長が短いためT-複雑度が低下したと考えられる。

表1 bzip2による圧縮率との相関係数

entropy	compress	gzip	C_T	I_T	H_T
0.13	0.91	0.98	-0.44	-0.44	0.64

```

case CompiledStatement.DELETE :
    return executeDeleteStatement(cs);

case CompiledStatement.CALL :
    return executeCallStatement(cs);

case CompiledStatement.DDL :
    return executeDDLStatement(cs);

```

図1 コード片1 (C_T : 51.09, H_T : 0.73, bzip2: 0.50)

```

// -----
// required
// -----
addColumn(t, "PROCEDURE_CAT", Types.VARCHAR);
addColumn(t, "PROCEDURE_SCHEM", Types.VARCHAR);
addColumn(t, "PROCEDURE_NAME", Types.VARCHAR, false); // not null

```

図2 コード片2 (C_T : 66.34, H_T : 0.75, bzip2: 0.41)

```

if ((bufpos + 1) >= len)
    System.arraycopy(buffer, bufpos - len + 1, ret, 0, len);
else
{
    System.arraycopy(buffer, bufsize - (len - bufpos - 1), ret, 0,
        len - bufpos - 1);
    System.arraycopy(buffer, 0, ret, len - bufpos - 1, bufpos + 1);
}

```

図3 コード片3 (C_T : 62.91, H_T : 0.81, bzip2: 0.47)

```

boolean    hasReturnValue;

outlen = parameters.length;
offset = 0;

```

図4 T-複雑度が最も低いコード片 (C_T : 37.91, H_T : 1.48, bzip2: 1.09)

4. おわりに

本稿では、T-複雑度によるソースコード中の類似文字列の検出の可能性を実験により評価した。実験結果より類似文字列検出に関して、T-平均情報量は bzip2 に近い傾向があることが確認できた。今後、bzip2 と T-平均情報量の検出結果の差が生じる要因の分析、類似文字列の位置の特定方法など研究を進める予定である。

参考文献

- 1) Titchener, M. R: Digital Encoding by Means of New T Codes to provide Improved Data Synchronization and Message Integrity, IEE Proceedings, Computer Digital Technology, Vol.131, No.4, pp.151-153 (1984).
- 2) U. Guenther: T-Complexity and T-Information Theory – an Executive Summary, CDMTCS Report 149, Centre for Discrete Mathematics and Theoretical Computer Science, The University of Auckland (2001).
- 3) M.R.Titchener: Deterministic computation of complexity, information and entropy, Proceedings of 1998 IEEE International Symposium on Information Theory, pp.326 (1998).
- 4) C言語によるT-複雑度などの計算プログラム“tcalc.c”
<http://tcode.auckland.ac.nz/~mark/>
- 5) 二村阿美, 門田暁人, 玉田春昭, 神崎雄一郎, 中村匡秀, 松本健一: 命令の乱雑さに基づくプログラム理解性の評価, ソフトウェア工学の基礎 XIX, 日本ソフトウェア科学会 FOSE2012, pp. 151-160 (2012).
- 6) Raymond P.L. Buse, Westley R. Weimer: Learning a Metric for Code Readability, IEEE Transactions on Software Engineering, Vol.36, No.4, pp.546-558 (2010).