修士学位論文

題目

LSH アルゴリズムを利用した類似ソースコードの高速検索手法

指導教員
井上 克郎 教授

報告者
川満 直弘

平成 28 年 2 月 9 日

大阪大学 大学院情報科学研究科
コンピュータサイエンス専攻 ソフトウェア工学講座

平成 27 年度 修士学位論文

LSH アルゴリズムを利用した類似ソースコードの高速検索手法

川満 直弘

内容梗概

ソフトウェアの開発において，他プロジェクトで開発したソフトウェアの再利用は頻繁に行われている．しかし，再利用されたソースコードの中には管理が行き届いていないものもあり，不具合を含んだソースコードを使い続けているものも多い．管理のためには出自に関する情報が必要であるが，そのような情報が失われている場合もある．そのような場合，失った情報を復元するための糸口の一つとなるのが再利用したソースコードの内容である．

本研究では，与えられたソースファイルに対し，類似するソースファイルを高速に検索する手法を提案する．内容が同一のものだけでなく類似するものを含めて出力することで，再利用の際に変更が加えられているソースファイルにも対応する．また，大量のソースファイルの中から検索を行うため，locality-sensitive hashing を利用することにより，高速な検索を可能にする．この手法によって，再利用したコードの出自を知ることが可能になる．

手法の妥当性の評価のために，再利用されている C 言語で記述されたライブラリを対象にケーススタディを行った．対象の中にはプロジェクト独自の変更が加えられ，再利用元ファイルと一致しなくなっているものも含まれていた．その結果，92%の精度で 200 プロジェクトの中から再利用元のソースファイルを検出することができた．また，4 ファイルを使用し検索の実行時間を測定したところ，各ファイルについて 1 秒以内で検索を完了した．

主な用語

ソフトウェア再利用
locality-sensitive hashing
類似検索

目 次

1　はじめに

ソフトウェアの開発において，他プロジェクトで開発したソフトウェアの再利用が頻繁に行われている．プロジェクト独自の変更が必要な場合やコンパイル手順を簡単にしたい場合には，再利用元プロジェクトのソースコードをコピーして取り込む，という方法で再利用が行われる．このような再利用には利点がある一方，再利用したコードに含まれる不具合を取り込んでしまう虞がある．再利用したコードを管理するためには再利用したコードがどのプロジェクト，バージョンのものなのかに関する情報が必要である．しかしながら，すべてのプロジェクトにおいてそのような情報が記録されているわけではないことが判明している [20].

再利用したソースファイルに対し，もしファイルの内容が一致するものが他のプロジェクトに存在するならば，そのプロジェクトに由来する可能性があるといえる．しかし，再利用の際にコードに変更が加えられることがあり，その場合にはソースファイルが単純に一致するかどうかを判定する，という方法では不十分である．出自を求める既存研究として，Inoue ら [8] は Ichi Tracker というシステムを提案した．このシステムは，コード片をクエリとし，コード検索エンジンを利用して検索を行う．また，我々のグループでは，リポジトリ中のソースコードを対象に，バージョンを検出する手法を提案した [12]．この手法では再利用元のリポジトリを用意しなければならない．

本研究では，与えられたソースファイルに対し，類似するソースコードを高速に検索する手法を提案する．検索の高速化のために，locality-sensitive hashing[7] を利用する．提案手法では，まず検索の対象となるソースファイルをデータベースに登録する．この際，LSH アルゴリズムを利用するために，ソースファイルを MinHash[2] の値からなるベクトルに変換する．検索を行う際には，同様に対象のソースファイルをベクトルに変換し，そのベクトルを用いて検索を行う．

手法の評価のために，2 ケースを利用して手法の妥当性を評価した．さらにケーススタディとして実際に再利用されたコードに対し手法を適用し，記録されている再利用元が検出できるか評価を行った．また，手法に要する時間についても評価した．

以降，2 章では研究の背景について述べる．3 章では本研究での提案手法について述べ，4 章では本研究においての実装について述べる．5 章では行ったケーススタディについて述べ，6 章で妥当性への脅威について述べる．最後に，7 章を本研究のまとめとする．

2　背景

本章では，研究の背景としてまずソースコードの再利用に関して述べる．その後，本研究で提案する手法の中で主要なアルゴリズムである，locality-sensitive hashing について述べる．

2.1　ソースコードの再利用

開発者はソフトウェアを開発する際に，他プロジェクトで開発したソフトウェアを再利用しながら開発を行っている．Heinemann ら [6] はオープンソースの Java のプロジェクトを対象に調査を行い，バイナリでの再利用である black-box reuse がソースコードの再利用である whilte-box reuse より優勢であったことを報告した．また，Rubin ら [18] は企業内において，開発したソフトウェアから新しい製品の開発に利用するために，再利用を行っていることを報告している．

再利用をすることによって開発のコストを抑え，ソフトウェアを効率的に開発することが可能になる．また，再利用されたコンポーネントはそうでないものより不具合が少ないことが Mohagheghi らによって報告されている [15]．したがって，再利用によって不具合が減る可能性があると考えられる．

開発者が再利用を行う際に，ソースコードを開発中のプロジェクトに取り込む，という方法がとられることがある．この方法がとられる理由として，例えばプロジェクト独自の変更が必要な場合が挙げられる．例えば，v8monkey[1] では，APNG(Animated Portable Network Graphics) フォーマットに対応するための修正を libpng[2] のライブラリに加えている．

ソースコードの再利用は有用である一方，バグやセキュリティに関する脆弱性を抱えているものを使用しないよう，開発者は注意しなければならない．もし再利用したソースコードに不具合が発見され，その修正が公開された場合，その修正を手元のソースコードにも適用するのが望ましい．しかしながら，そのためには再利用したソースコードが管理されている必要がある．Xia ら [20] はオープンソースのライブラリのコードを再利用しているプロジェクトについて，使用しているライブラリが脆弱性を抱える可能性のあるバージョンであるかどうかを調査した．その結果，調査対象としたプロジェクトのうち，zlib を利用しているプロジェクトでは 31.1%，libcurl を利用しているプロジェクトでは 85.7%，libpng を利用しているプロジェクトでは 92%のプロジェクトが，脆弱性を抱える可能性のあるバージョンを利用していることが判明した．さらに，18.7%のプロジェクトには使用したライブラリについて，どのバージョンを利用したかに関する情報が残っていなかった．加えて，4.9%のプロ

[1] https://github.com/zpao/v8monkey
[2] http://www.libpng.org/pub/png/libpng.html

ジェクトではディレクトリ名が変更されたり，再利用したソースコードに他のソースコードが混ぜられ，管理することが難しい状態になっていることが判明した．このような管理のされていないプロジェクトに対して，再利用元を特定することは有用であると考えられる．

ソフトウェアの再利用の検知は，バイナリを対象としたものやソースコードを対象としたものがある．バイナリを対象としたものとして，Davies ら [4, 5] は Java のバイナリから，クラス中のシグネチャを利用する手法を提案した．Sæbjørnsen ら [19] は実行ファイルのバイナリからのコードクローン検出手法を提案した．Qiu ら [17] は，バイナリからライブラリの関数を特定する手法を提案した．

Inoue ら [8] は，Ichi Tracker というシステムを提案した．これは，コードの断片などをクエリとし，そのコード片のクローンを含むソースコードをコード検索エンジンを使用して検索する．システムの出力として見つかったクローンとともに，各クローンについてそのクローンの最終変更日時，どのプロジェクトのコードか，どの程度クエリの内容を含んでいるか，などの情報が得られる．

我々の研究グループ [12] ではリポジトリに含まれているソースコードについて，バージョンを推定する手法を提案した．ソースコードの類似度として最長共通部分列に基づいた類似度 [11] を使用し，もっとも類似度の高いものが再利用元であるという仮定に基づいてバージョンを提示した．この手法には入力として再利用を行ったリポジトリとその再利用元のリポジトリが必要なため，再利用元が不明な場合は利用することができない．

2.2 Locality-Sensitive Hashing による検索

本研究ではソースファイルの再利用を検出するという問題を，クエリとして与えられたファイルに類似したソースファイルを見つける問題の一種ととらえる．単純にクエリと検索対象の 1 件 1 件を比較する場合，検索対象の数に比例した計算時間が必要になる．しかし，出自を調べるためには複数のプロジェクトのソースファイルを検索対象にする必要があり，検索対象の数が大きくなることが予想される．したがって，単純にクエリとの比較を行う方法をとった場合，許容できないほどの実行時間がかかる可能性がある．そこで，本研究では高速化のために locality-sensitive hashing(以下 LSH とも表記) を用いる．LSH は近似最近傍検索や，クラスタリングに用いられるアルゴリズムである [7]．LSH は以下の 2 つの性質を持つような関数族 F を用いる．

$$\text{if } d(x,y) \leq d_1 \text{ then } \Pr[f(x) = f(y)] \geq p_1$$

$$\text{if } d(x,y) \geq d_2 \text{ then } \Pr[f(x) = f(y)] \leq p_2$$

ただし，$d_1 < d_2$ は距離尺度 d による距離，f は関数族 F 中の関数である．また，$p_1 > p2$ を満たす．このような関数族 F は，(d_1, d_2, p_1, p_2)-sensitive family と呼ばれる．関数族 F

中の十分な数の f を利用して，$f(x) = f(y)$ かどうかの結果を組み合わせることで，x, y の距離が近いかどうかを高い確率で判定することができる．

LSH は複数の適用例がある．Manku ら [14] は，Web でのクローリングにおいて，内容がほぼ重複している Web ページを検出するための手法を提案した．Das ら [3] は，Google News のために協調フィルタリングを使用し，ユーザの行動から記事を推薦する手法を提案した．Brinza ら [1] は，ゲノムワイド関連解析において遺伝子座のクラスタリングのために利用した．Jing ら [10] は，画像検索のために LSH を利用している．

LSH はソースコードにおいても利用されている．Jiang ら [9] らは，類似する部分木を高速に検出する手法をソースコードに用いることでクローンの検出を実現しているが，部分木の特徴ベクトルに対しクラスタリングを行うために LSH を利用した．山中ら [21] は TF-IDF 法を特徴量とし，クラスタリングを行ってタイプ 4 の関数クローン検出の手法を提案したが，高速な検出を行うために特徴ベクトルのクラスタリングに LSH を利用している．

本研究では，ソースファイルの類似度に n-gram および Jaccard 係数を利用した類似度を用いる．TF-IDF 法と比較すると，Jaccard 係数を利用することには類似度を求める対象の 2 ファイルのみから類似度を求めることができるという利点がある．

LSH に用いることのできる尺度の例としては，以下のようなものがある [7, 16]．

- ハミング距離
- Jaccard 係数
- コサイン類似度
- earth mover's distance

LSH に Jaccard 係数を尺度として用いる場合，MinHash[2] が用いられる．MinHash は Web サービス中で類似文書を検出するために利用された．MinHash は，与えられた集合の要素すべてに対しハッシュ値をとり，その最小値を利用して集合間の類似度を推定する手法である．

3　提案手法

本研究では，検索のクエリとして与えられたソースファイルに対し，大量のソースファイルの中からソースコードの内容がクエリに類似するものを検索する手法を提案する．LSH を用いることにより，高速な検索を可能にする．

本手法ではさらに，検索された結果に対して類似度の推定値を提示する．この推定値は効率的に求めることができ，検索結果に含まれる各ソースファイルについて，推定値ではあるがクエリとの類似度を個別に知ることができる．

手法は大きく分けて，検索する対象となるソースファイルをデータベースに登録するステップ，クエリとなるソースファイルに対してデータベースから類似ソースコードを検索するステップの 2 つからなる．まず本手法で用いるソースファイル間の類似度を定義し，MinHash，LSH について説明したのち，これらのステップについて説明する．

3.1　ソースファイル間の類似度の定義

本節では，本研究で用いるソースファイル間の類似度について述べる．2 つのソースファイルの類似度は，n-gram と Jaccard 係数を利用した類似度である．この類似度を用いる利点として，関数単位の並べ替えなどの影響を過度に受けないという利点がある．

具体的な類似度を求める手順は，以下のステップからなる．

1. ソースファイルの字句分割
2. 字句列からの n-gram の抽出
3. n-gram の多重集合からの集合への変換
4. 集合に対する Jaccard 係数の算出

以降，それぞれのステップについて述べる．以下では類似度を求めるソースファイルを s, t とする．

字句分割

ソースファイル s からコメントを取り除き，字句の列に分割し，T_s を得る．コメントを取り除くことで，コメント量の大小や，コメントの違いが類似度に影響を与えることを防ぐ．

ソースファイル t にも同様の操作を行い，字句列 T_t を得る．

n-gram の抽出

字句列 T_s から長さ n の部分文字列を取り出すことにより，n-gram の多重集合 M_s へ変換する．

たとえば，$n = 3$ のとき，要素列 $e_1, e_2, e_3, ..., e_m$ に対して得られる n-gram は，$\{\{(\$, \$, e_1), (\$, e_1, e_2), (e_1, e_2, e_3), (e_2, e_3, e_4), ..., (e_{m-1}, e_m, \$), (e_m, \$, \$)\}\}$ となる．ただし，要素$は要素列の先頭および末尾の要素に付加するダミーの要素である．先頭及び末尾にダミーの要素を付加することによって，列のすべての要素が n 個の n-gram に含まれる．

同様にして，字句列 T_t から多重集合 M_t を得る．

多重集合からの集合への変換

多重集合 M_s を集合 S_s に変換することを目的として，多重集合の各要素に番号を付加することによって集合に変換する．各要素に付加する番号は，ある要素 e について，変換前の多重集合に要素 e が k 個含まれている場合，各要素に 1 から k の値が付加される．

たとえば，要素 a, b, c からなる多重集合 $\{\{a, a, b, b, b, c\}\}$ をこの手法により集合に変換した場合，得られる集合は $\{(1, a), (2, a), (1, b), (2, b), (3, b), (1, c)\}$ となる．

同様にして，多重集合 M_t から集合 S_t を得る．ただし，付加する番号は M_s に依存せず，M_s と M_t に同じ要素が含まれていたとしても，M_t から S_t に変換する過程で付加する番号は 1 から順に与える．

Jaccard 係数の算出

集合 S_s と集合 S_t について，以下の式を用いて Jaccard 係数 $\mathrm{Jaccard}(S_s, S_t)$ を算出する．

$$\mathrm{Jaccard}(S_s, S_t) = \frac{|S_s \cap S_t|}{|S_s \cup S_t|}$$

この $\mathrm{Jaccard}(S_s, S_t)$ の値を，ソースファイル s, t の類似度とする．

3.2 Locality-Sensitive Hashing

locality-sensitive hashing は類似検索に利用される手法の一つである [7]．本研究では，以下の式で表される関係が成り立つハッシュ関数の族 F を利用して LSH を構成する [16]．

$$\mathrm{Pr}_{h \in F}[h(x) = h(y)] = sim(x, y)$$

ただし，$sim(x, y)$ は x, y 間の類似度であり，$0 \leq sim(x, y) \leq 1$ である．類似度として Jaccard 係数を用いる場合，F として MinHash が利用可能である．

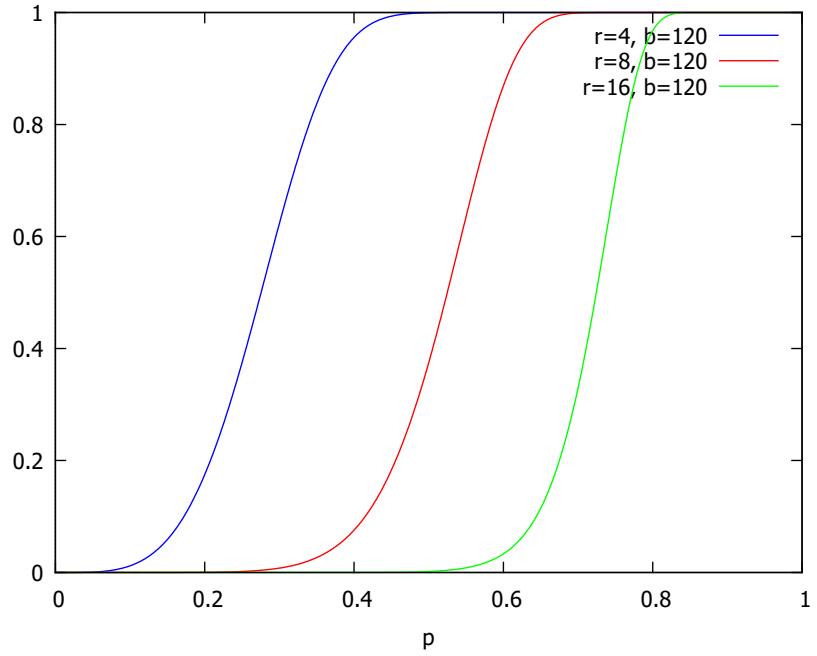

図 1: LSH のパラメータの違いが同じバケットに入る確率に与える影響

MinHash は，集合間の類似度を高速に推定する手法である．集合中の各要素についてハッシュ値を求め，その中の最小値を求める．同様の操作を別の集合に対しても行った時，それら 2 つの最小値が一致する確率は，2 つの集合に対する Jaccard 係数に等しい．

文献 [13] の方法を用いて LSH を構成する．2 集合について考える．ハッシュ関数を $r \times b$ 個用意し，各集合について r 次元の MinHash の値からなるベクトルを b 個求める．2 集合間において，各ハッシュ関数に対応するベクトル同士を比較する．この時，2 集合間での MinHash の値が一致する確率，すなわち Jaccard 係数を p とすると，b 個のベクトルのうち 1 つ以上のベクトルが一致する確率は，$1-(1-p^r)^b$ で表される．これらのベクトルをハッシュテーブルのキーとし，ベクトルが一致した場合に同じバケットに入っていると扱うことにより，類似度 p の要素同士が確率 $1-(1-p^r)^b$ で同じバケットに入る，ということを実現する．

図 1 に，実際にパラメータの値を与えた際の $1-(1-p^r)^b$ のグラフを示す．パラメータは $b=120$ であり，r については $r=4, r=8, r=16$ の 3 通りである．図に示すとおり，パラメータを調整することで，同じバケットに入る確率を調整することができる．

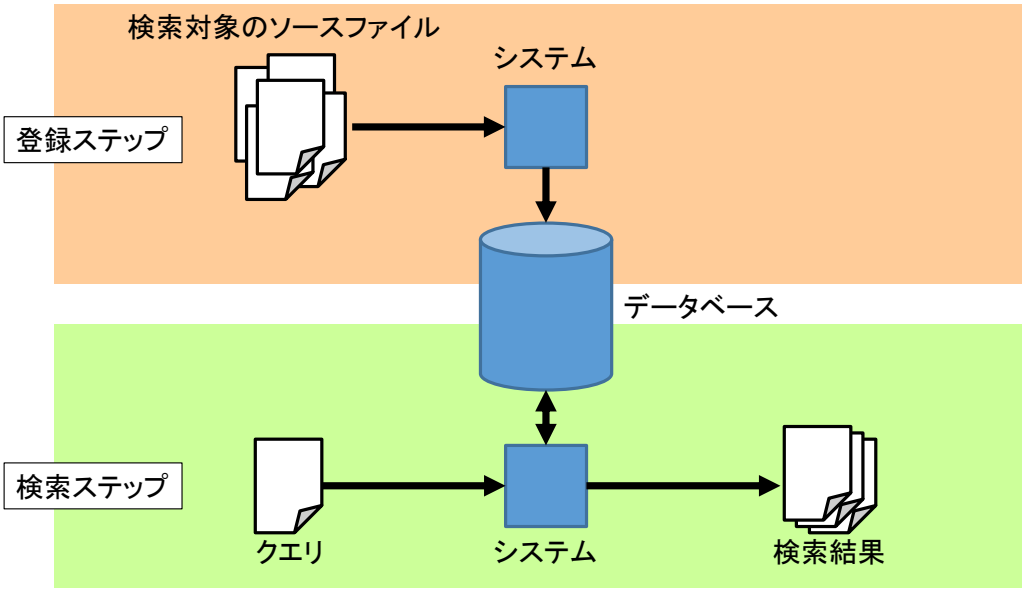

図 2: 提案手法の概要

3.3 LSH を利用したソースファイルの検索

本節では，データベース上で LSH を利用した検索を行う方法について説明する．データベースを用いることにより，検索対象のファイル数が大きく，必要なデータすべてを主記憶に収めることができない場合にも対応する．

提案手法の概要を図 2 に示す．手法は 2 つのステップからなる．登録ステップでは複数のソースファイルを入力し，データベースに登録する．検索ステップでは，クエリとなるソースファイル q を与え，検索結果である類似ファイルの集合に加え，それぞれの結果のファイルに対して，ソースファイル q との類似度の推定値を出力する．

3.3.1 検索対象のデータベースへの登録

このステップでは，検索の対象となるソースファイルを入力とし，データベースに登録する．

まず LSH を構成する MinHash において使用するハッシュ関数を $R \times B$ 個用意する．R, B

は LSH の検索パラメータ r, b の上限となる．用意するハッシュ関数は番号の付加された n-gram からハッシュ値への関数であり，それらを h_i とする．ただし，$1 \leq i \leq R \times B$ である．次に検索の対象となるソースファイルをデータベースに登録するが，登録する情報はソースファイル 1 件に対し，各ハッシュ関数から求められる MinHash の値もデータベースに登録する．h_i を用いてソースファイルから MinHash の値を求める関数を m_i と表す．ソースファイル f に対し，番号の付加された n-gram を g，ソースファイルから番号の付加された n-gram の集合への関数を G とすると，$m_i(f) = \min_{g \in G(f)} h_i(g)$ である．m_i を利用し，ソースファイル f に対し，$m_1(f), m_2(f), \ldots, m_{R \times B}(f)$ を同時に登録する．

この登録した MinHash の値を利用して LSH を構成し検索を行うため，検索の高速化のためにインデックスを作成する．R 個の MinHash の値を一組とし，B 組それぞれに対してインデックスを作成する．すなわち，$(m_1(f), m_2(f), \ldots, m_R(f))$ を 1 組としてインデックス I_1，$(m_{R+1}(f), m_{R+2}(f), \ldots, m_{R \times 2}(f))$ を 1 組としてインデックス I_2，というようにインデックスを構成し，合計 B 組のインデックスを構成する．

3.3.2 類似ソースファイルの検索

このステップでは，与えられたソースファイルに類似するソースファイルを，データベースから検索する．検索には，クエリとなるソースファイルに加え，LSH のパラメータ $1 \leq r \leq R$, $1 \leq b \leq B$ を与える．与えられたソースファイルに対して MinHash の値を求め，その値から検索を行う．$1 \leq i \leq b$ について，インデックス I_i を用いて，

$$S_i(q) = \{f | \forall j \in [1, r]. m_{j+R \times (i-1)}(q) = m_{j+R \times (i-1)}(f)\}$$

なる S_i を求める．この集合 S_i の和集合，すなわち $\bigcup_{1 \leq i \leq b} S_i(q)$ が検索の結果，つまりクエリと類似するソースファイルの集合である．

また，検索結果に含まれるそれぞれのソースファイルに対して，クエリとの類似度の推定値を併せて出力する．類似度は最尤推定を用いて推定する．

検索結果中のある類似ソースファイルについて，集合 $S_1, S_2, ..., S_b$ 中の x 個の集合に含まれているとする．類似度が p で表される時，r 次元のベクトルについて x 個のベクトルが一致し，$b - x$ 個不一致となる確率，p に対する尤度関数 $L(p)$ は

$$L(p) = (p^r)^x (1 - p^r)^{b-x} \binom{b}{x}$$

で表される．したがって，類似度 p に対する最尤推定量 $\hat{p}$ は，$0 \leq p \leq 1$ であることを踏まえると，

$$\hat{p} = \sqrt[r]{\frac{x}{b}}$$

となる．ここに示す通り，類似度の推定値がとりうる値は，$x \geq 1$ の場合，b 通りに限定される．図 3 に，$b = 120, 1 \leq r \leq 8$ の場合の推定値のとる値を示す．横軸が推定値の値，縦軸が各 r の値を表す．$b = 120$ の場合に限らず，$r = 1$ の場合は推定値としてとりうる値の間隔は等しく，MinHash を用いて Jaccard 係数を推定する方法そのものである．一方，$r > 1$ の場合，推定値のとりうる値の間隔は 1 に近づくにつれ狭くなり，1 に近いほど細かく，0 に近いほど粗くなる．

推定量の偏り $B(\hat{p})$ は

$$B(\hat{p}) = \sum_{x=0}^{b} \sqrt[r]{\frac{x}{b}}(p^r)^x(1-p^r)^{b-x}\binom{b}{x} - p$$

となる．この推定量は，$r = 1$ の場合 $B(\hat{p}) = 0$ となり，不偏推定量である．$r > 1$ の場合の例として，パラメータが $b = 120$ であり，$r = 2, r = 4, r = 8$ の場合の推定量の偏りについて p の値を変えながら図示したのが，図 4 である．横軸は p の値であり，縦軸は偏りである．図示した 3 通りのパラメータでは，偏りが負となっており，実際の類似度よりも低い値として推定される傾向があることを示している．また，p の値が 1 付近に近づくと偏りが 0 に近づくことを示している．

推定量の分散 $Var(\hat{p})$ は

$$Var(\hat{p}) = \sum_{x=0}^{b} \left(\sqrt[r]{\frac{x}{b}}\right)^2 (p^r)^x(1-p^r)^{b-x}\binom{b}{x} - \left(\sum_{x=0}^{b} \sqrt[r]{\frac{x}{b}}(p^r)^x(1-p^r)^{b-x}\binom{b}{x}\right)^2$$

となる．特定のパラメータの例として，図 5 にパラメータが $b = 120$ であり，$r = 2, r = 4, r = 8$ の場合の推定量の分散について p の値を変えながら図示した．横軸は p の値であり，縦軸は分散である．分散についても，p の値が 1 付近に近づくと分散が 0 に近づくことを示している．

推定量の平均二乗誤差 $MSE(\hat{p})$ は，

$$MSE(\hat{p}) = Var(\hat{p}) + (B(\hat{p}))^2$$

である．同じく例として図 6 にパラメータが $b = 120$ であり，$r = 2, r = 4, r = 8$ の場合の推定量の平均二乗誤差について p の値を変えながら図示した．横軸は p の値であり，縦軸は平均二乗誤差である．平均二乗誤差についても偏りや分散と同じく，p の値が 1 付近で 0 に近づいている．

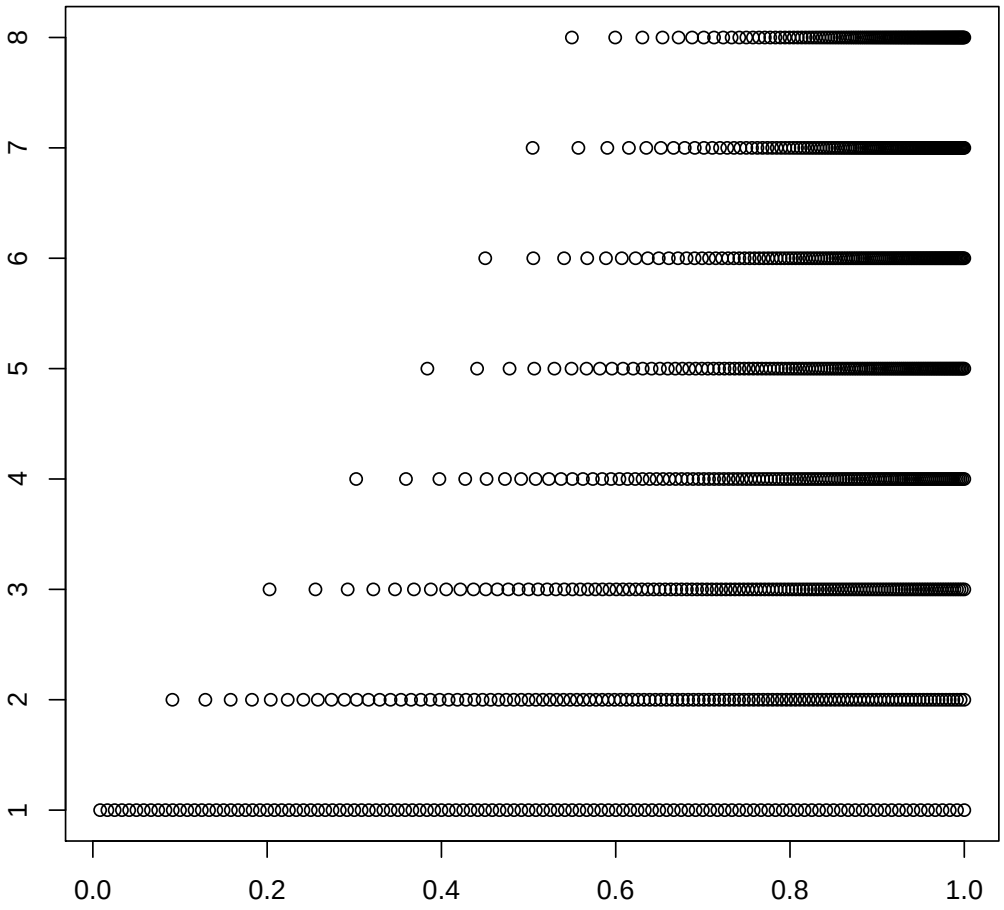

図 3: $b = 120$ の場合の類似度の推定値のとる値

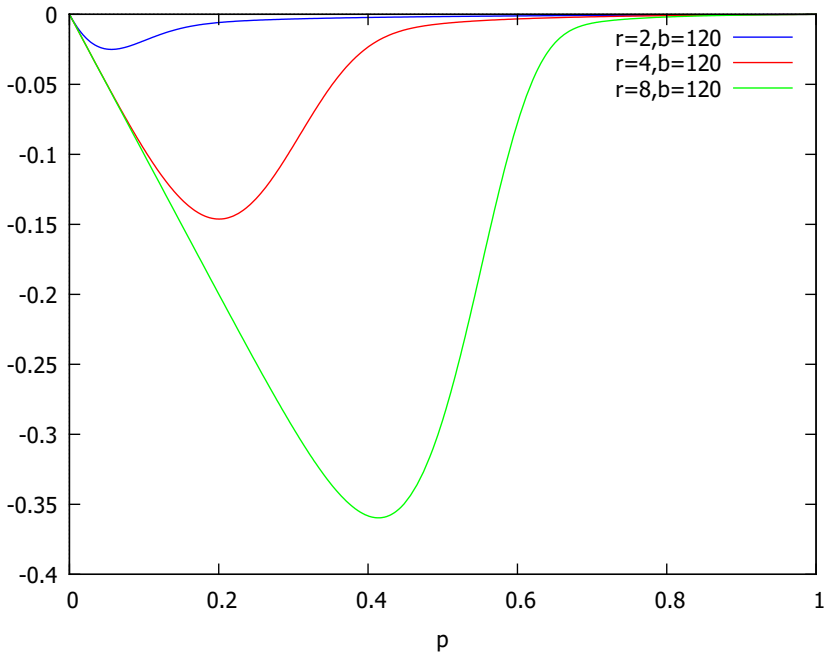

図 4: $b = 120$ の場合の推定量の偏り

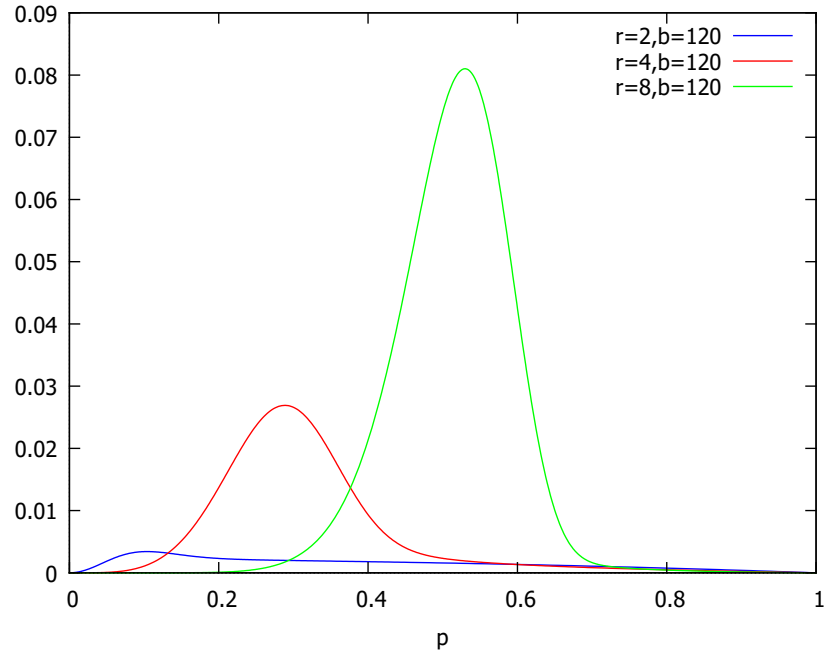

図 5: $b = 120$ の場合の推定量の分散

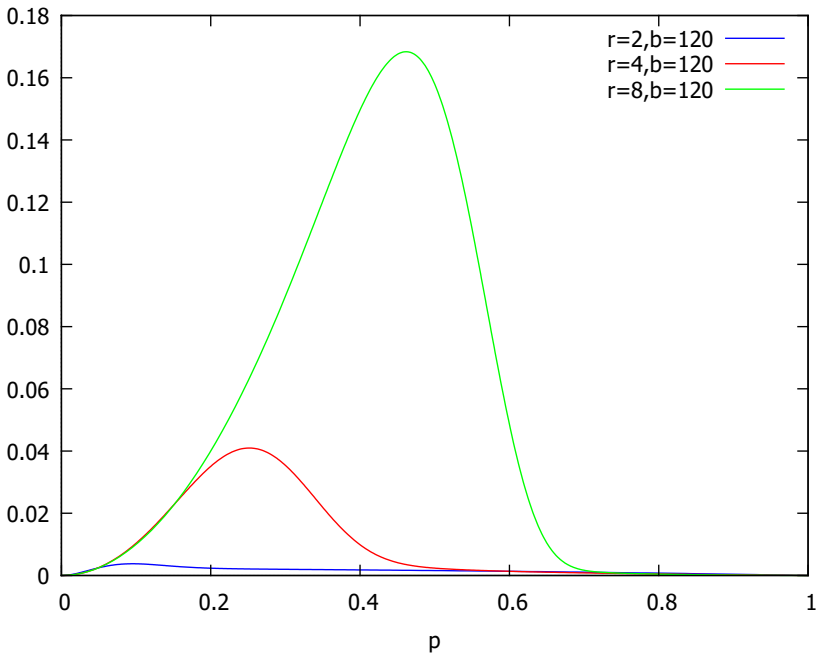

図 6: $b = 120$ の場合の推定量の平均二乗誤差

4 実装

4.1 類似度について

手法の対象を C 言語のソースファイルとし，類似度で用いる n-gram の n の値として，$n = 3$ とした．

異なるファイル間での類似度と，同じファイルの別バージョン間での類似度の違いについて調査した．調査には Ubuntu の apt で手に入るソースコード，10 パッケージ，505 ファイルを対象とした．対象の内訳について，表 1 に示す．対象となったファイルすべての組み合わせの間で類似度を求めた．

図 7 に類似度に対する頻度を示す．左図はパッケージが異なり，かつファイル名も異なるファイル間での類似度を表しており，右図は同一パッケージ，同一パス，同一ファイル名であり別の tar ファイル，つまりバージョンが異なるファイル間の類似度を表している．左図には 113735 件の類似度が含まれており，右図には 245 件の類似度が含まれている．ただし，右図中で 101 件について，類似度が 1.0 であった．

この結果より，3-gram を使用した類似度は二つのファイルが同一ファイルの別バージョンか否かの判別に有効と考えられる．

表 1: データセット

パッケージ	tar ファイル	.c ファイル数	.h ファイル数
original-awk	original-awk_2012-12-20.orig.tar.gz	10	3
original-awk	original-awk_2011-08-10.orig.tar.gz	10	3
mgcv	mgcv_1.8-4.orig.tar.gz	13	5
mgcv	mgcv_1.7-12.orig.tar.gz	11	6
seaview	seaview_4.3.1.orig.tar.gz	10	8
seaview	seaview_4.5.3.1.orig.tar.gz	12	11
fglrx-installer	fglrx-installer_8.960.orig.tar.gz	11	24
fglrx-installer	fglrx-installer_15.200.orig.tar.gz	11	24
fglrx-installer-updates	fglrx-installer-updates_15.200.orig.tar.gz	11	24
fglrx-installer-updates	fglrx-installer-updates_8.960.orig.tar.gz	11	24
foreign	foreign_0.8.62.orig.tar.gz	14	12
foreign	foreign_0.8.48.orig.tar.gz	14	12
libhdhomerun	libhdhomerun_20140604.orig.tar.gz	14	15
libhdhomerun	libhdhomerun_20120128.orig.tar.gz	14	15
r-cran-eco	r-cran-eco_3.1-4.orig.tar.gz	17	7
r-cran-eco	r-cran-eco_3.1-6.orig.tar.gz	17	7
r-cran-xml	r-cran-xml_3.6-2.orig.tar.gz	17	8
r-cran-xml	r-cran-xml_3.98-1.1.orig.tar.gz	19	9
rt-tests	rt-tests_0.89.orig.tar.gz	20	7
rt-tests	rt-tests_0.83.orig.tar.gz	18	7

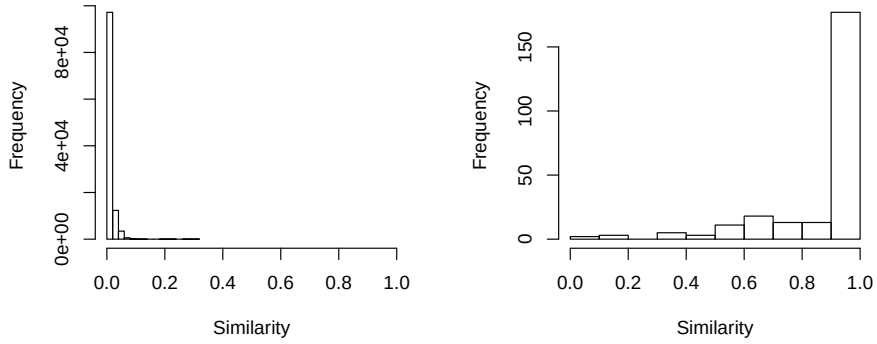

図 7: 類似度の分布

4.2 ハッシュ長について

MinHash のハッシュとして，64bit 長のハッシュ値を使用した．MinHash の「ハッシュ値の最小値をとる」という操作により，MinHash の値は小さい値ほど出やすくなり，ハッシュ値の値域の中での最小値がもっとも出やすい値となる．ある集合の MinHash の値が，ハッシュ値の値域の最小値となる確率について示す．

ハッシュ長を b bit，集合の要素数を n とする．また，ハッシュ関数は完全にランダムにふるまうとする．このとき，MinHash の値が値域の最小値となる確率 $P(b,n)$ は

$$P(b,n) = 1 - \frac{(2^b-1)^n}{(2^b)^n}$$

となる．n はおよそソースファイルに含まれる字句数に対応する．例えば $n = 1000000$ とした場合の $P(b,n)$ の値は

$$P(64, 1000000) = 5.4 \times 10^{-14}$$

となる．MinHash の値が小さな値ほど出やすいことから，64bit 長のハッシュを利用することで意図しない衝突の起きる確率は十分に低いと考えられる．

4.3 ハッシュ関数について

提案手法では番号の付加された n-gram に対するハッシュ関数を複数用意する必要がある．今回は，簡便な方法として以下のような実装を行った．

番号の付加された 3-gram(num, t_1, t_2, t_3) に対するハッシュ関数 h_i によるハッシュ値は，

$$(((num \times 65537) + t_1) \times 65537 + t_2) \times 65537 + t_3) \times a_i + b_i$$

によって求める．ただし，num は 3-gram に付加された番号，t_1, t_2, t_3 は n-gram の要素，つまり字句に対する Java の hashCode メソッドの値，a_i, b_i はそれぞれハッシュ関数ごとに用意する定数である．演算は 64bit で行われ，演算中のオーバーフローは下位 64bit が残される．a_i は 64bit 乱数と 1 とのビット論理和をとることによって，b_i は 64bit 乱数によって用意した．

5　ケーススタディ

5.1　検索結果および類似度の推定値の評価

実際の検索において，クエリに対する類似度の高いものが検索の結果に表れ，類似度の低いものが結果に表れないということを確認するために，libpng および libcurl に対して実験を行った．

5.1.1　libpng の png.c

libpng のリポジトリ [3] 中に含まれる，すべてのリビジョンの c ファイル，h ファイルをデータセットとし，実験を行った．具体的には，`git rev-list --all --objects` によって得られたリストの中から，.c ファイルおよび.h ファイルを対象とした．対象のファイル数は 20977 である．libpng に含まれ，git でのファイルハッシュの値が 59d747d9947ad43c4354f535ad58d5bbd06a1dfd である，`v1.6.0` のタグがついているコミットのファイル png.c をクエリとして検索を行った．検索のパラメータは，$r = 8, b = 120$ である．

図 8 に検索の結果をヒストグラムとして示す．この図は，縦軸にデータセットに含まれる件数，横軸にクエリとの類似度を示している．青で示される部分はデータセット中から検索の結果に表れた件数を示しており，赤で示される部分はデータセット中に含まれているが，検索の結果に表れなかったものを示す．クエリとの類似度が 0.2 未満の件数については，件数が非常に大きくなるために図に表示していない．

検索の結果に表れたファイル数は 716 であり，これはデータセット全体の 3.4%である．検索の結果に表れなかったものの中での類似度の最高値は 0.482 であり，それより類似度の高いものはすべて検索の結果に表れた．また，検索に表れたものの中で類似度の最低値は 0.598 であり，それより類似度の低いものは全く検索の結果に表れなかった．

図 9 に，検索結果中のファイルとクエリ間の類似度について，推定値と実際の値との関連を示す．x 軸は類似度の推定値，y 軸は実際の類似度を表す．

5.1.2　libcurl の url.c

libcurl のリポジトリ [4] に含まれる，すべてのリビジョンの c ファイル，h ファイルをデータセットとし，実験を行った．対象のファイル数は 23273 である．libcurl に含まれ，ハッシュ値が 42bf1eb1d0cc17e020c89390e4ee967bcf27e34e である，`curl-7_47_0` のタグがついているコミットの url.c をクエリとして検索を行った．

[3] `git://git.code.sf.net/p/libpng/code`
[4] `https://github.com/bagder/curl.git`

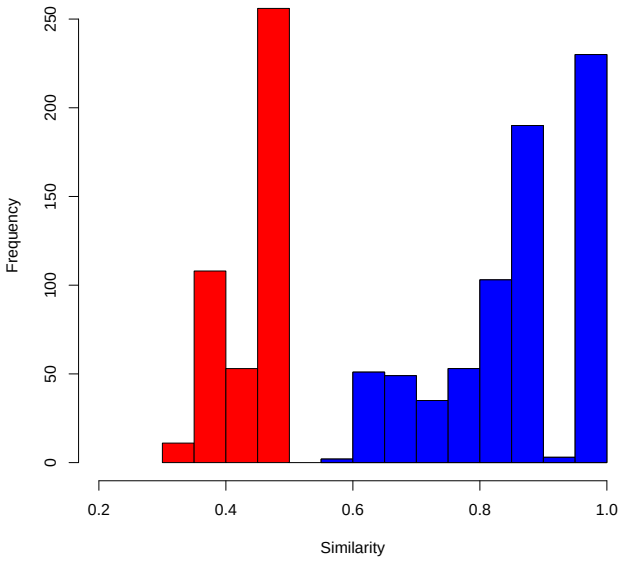

図 8: ファイルハッシュの値が 59d747 の png.c をクエリとした際の検索結果

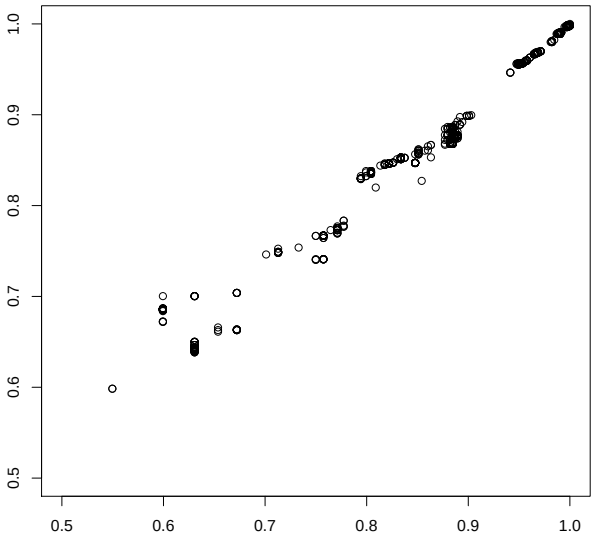

図 9: png.c をクエリとした際の類似度の推定値と実際の類似度

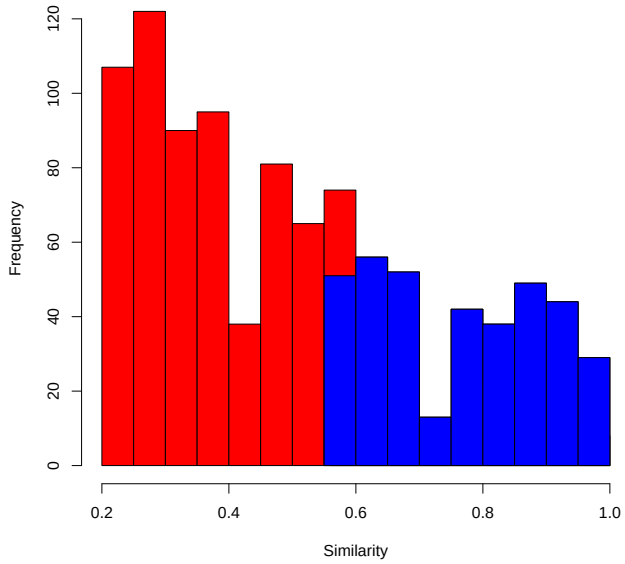

図 10: ファイルハッシュの値が 42bf1e の url.c をクエリとした際の検索結果

図 10 に検索の結果を示す．図の形式は図 8 と同様である．

検索の結果に表れたファイル数は 374 ファイルであり，これはデータセット全体の 1.6%である．検索の結果に表れなかったものの中での類似度の最高値は 0.578 であり，それより類似度の高いものはすべて検索の結果に表れた．また，検索に表れたものの中での類似度の最低値は 0.572 であり，それより類似度の低いものは全く検索の結果に表れなかった．

図 11 に，検索結果中のファイルとクエリ間の類似度について，推定値と実際の値との関連を示す．x 軸は類似度の推定値，y 軸は実際の類似度を表す．

5.2 ファイルの出自検索能力の評価

類似度の推定値を用いて再利用元のプロジェクト，ファイル，バージョンを特定することができるかケーススタディを行った．ケーススタディの対象として，v8monkey[5] 中に含まれる libpng のソースコードを利用した．v8monkey には複数のコミットメッセージに，libpng のどのバージョンに更新したかが記録されている．また，複数のソースコードに変更が加えられており，それらは Animated Portable Network Graphics フォーマットに対応するためと思われる．

検索の対象として，libpng プロジェクトを含む 200 プロジェクトのリポジトリを用意し

[5] `https://github.com/zpao/v8monkey.git`

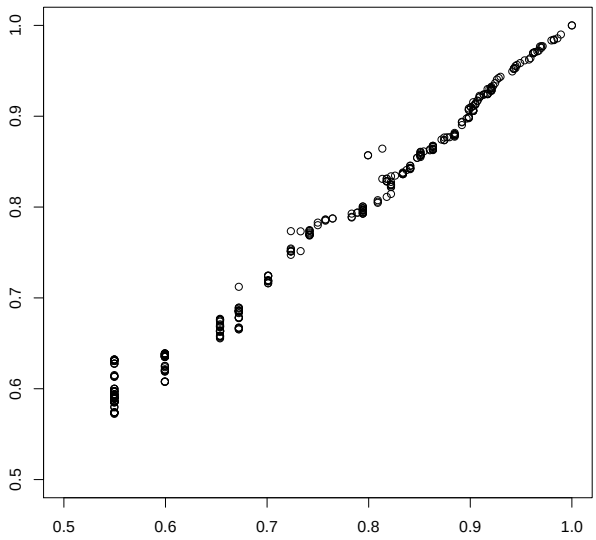

図 11: url.c をクエリとした際の類似度の推定値と実際の類似度

た．これらのプロジェクトは，GitHub において 2016 年 1 月 31 日に”lib language:c” というクエリを用いて検索を行い，その検索結果の上位 200 件のリポジトリを同年 2 月 1 日に得たものである．付録中の表 6 にプロジェクトの一覧を示す．

実験の手順として，まず検索対象のプロジェクト中の，履歴中に含まれるすべての.c ファイルおよび.h ファイルをデータベースに登録した．登録された件数は 567113 件である．次に v8monkey のコミットのうち libpng のどのバージョンにアップデートしたかが明記されているコミットを列挙した．この条件を満たすコミットは計 14 件見つかった．それらのコミットで新たに加えられた，もしくは変更されたファイルのうち，modules/libimg/png 以下に含まれるファイルをクエリとし，検索を行った．ただし，以下のファイルについては評価対象から除外した．

- コミット 5b7a4ae8d0d380ba0828c844634fc614da64e924，
  および 5d210c7d984190f89a4ad175909e01885324f1de に含まれる pngvcrd.c および pnggccrd.c．これらのファイルはファイル内のソースコードが削除され，コメントのみからなるファイルとなっていた．
- mozpngconf.h．このファイルは libpng に存在しないファイルであり，v8monkey の開発者が追加したものと思われる．

対象となったファイルは，18 ファイル，延べ 197 件である．このうちの 57 件については，

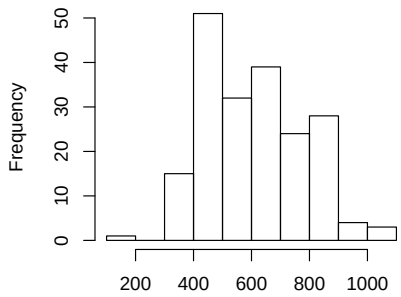

図 12: 各クエリで検索結果に表れるファイル数の分布

再利用元のファイルと内容が一致し，それ以外の 140 件は再利用元とファイルの内容が一致しなかった．検索のパラメータは，$r = 8, b = 120$ とした．

検索の結果，すべての結果について再利用元の該当するバージョンのファイルが検索結果に表れた．クエリ 197 件に対し，検索結果に表れたファイルの数の合計は，120001 件であった．各クエリで検索結果に表れるファイル数は，最低 167 件，最高 1031 件，平均 609 件であった．各クエリで検索結果に表れるファイル数の分布を，図 12 に示す．横軸は検索結果に表れるファイル数，縦軸はそのような検索結果となるクエリの頻度である．検索結果には libpng のファイル以外に，libpng を利用している 3 プロジェクト計 97 件含まれていた．

さらに，各クエリについて，類似度の推定値が検索結果に表れる類似度の推定値の中での最高値になるかどうかを調査した．結果の理解のために，2 つの基準から結果を 4 つに分類した．

基準 1 再利用元のバージョンの類似度の推定値が，検索結果に表れる類似度の推定値の中での最高値になる

基準 2 検索結果に表れたファイルの中で，再利用元のバージョンの類似度の真の値が，検索結果に表れた類似度の真の値の中での最高値になる

基準 2 の対象が検索結果に表れたファイルに限られているのは，時間の制約上データベースすべてのファイルに対し 197 件のクエリとの類似度の真の値を求めることができないためである．結果を基準によって分類した結果を表 2 に，コミットごとに分類した結果を表 3 に，ファイルごとに分類した結果を表 4 に示す．

クエリとなる対象の 197 件のうち 182 件は基準 1 を満たす，つまりクエリと記録されている再利用元との類似度の推定値が，検索結果に表れる類似度の推定値の最高値と等しくなっ

表 2: 分類の結果

	基準 1 を満たす	基準 1 を満たさない	計
基準 2 を満たす	177	10	187
基準 2 を満たさない	5	5	10
計	182	15	197

表 3: コミットごとの結果

コミット ID	対象ファイル数	基準 1 を満たす	基準 2 を満たす
04dd242a1f7c2fc33b295d2fd715adbc6b0b1ace	12	12	12
051d876b20fc709cb3558238a5deaf3e8949373f	17	17	17
3a04be0690dff135ec42784557fedbf6c572cd22	15	14	15
5b7a4ae8d0d380ba0828c844634fc614da64e924	14	13	13
5d210c7d984190f89a4ad175909e01885324f1de	14	12	12
6b63cb5ed0105c43668de3752f615bbabb1a7e95	10	8	10
943791bbcb7bd235b8b0fdd90a914074447dc0bb	10	9	9
9def47a86c95fd5fe3560d2b09b508ca639d27e5	13	13	12
a06f0a365e6ba83e2d09294cc1640ee1d4c0353b	17	17	16
a16fd7ff34a17594f61252f7f8159d8d6aff6f53	15	13	15
b7e2b4075f286be1204c0ae883adb0989559ba6b	17	15	16
e9ee3f8f7d6491700e0637df4b4389839fd6e960	17	15	16
f3231fdc19ee5c529c53f2570012a47e350cb1dc	18	16	16
fd5260880325ca83b758fc793ccf83b0216d7907	8	8	8
計	197	182	187

表 4: ファイルごとの結果

ファイル	対象コミット数	基準 1 を満たす	基準 2 を満たす
png.c	14	14	14
png.h	14	14	14
pngconf.h	14	9	13
pngerror.c	11	11	11
pnggccrd.c	1	1	1
pngget.c	12	12	11
pngmem.c	8	8	8
pngpread.c	12	12	11
pngpriv.h	5	3	5
pngread.c	12	11	12
pngrio.c	7	7	7
pngrtran.c	14	14	13
pngrutil.c	14	13	12
pngset.c	12	11	11
pngtrans.c	10	7	7
pngwio.c	8	8	8
pngwrite.c	12	12	12
pngwtran.c	6	6	6
pngwutil.c	11	9	11
計	197	182	187

た．基準 1 を満たすクエリについて，各クエリの検索結果中で類似度の推定値が最高値をとるファイル数は，最低 1 件，最高 56 件，平均 11 件であった．また，その分布を図 13 に示す．図の表し方は図 12 と同様である．さらに，基準 1 を満たすクエリの各検索結果に含まれる類似度の推定値が最高値をとるファイルを対象に，コメント，フォーマットの違いを無視して重複を取り除いた数を求めた．ただし，重複の検出には md5 チェックサムを用いた．その結果，重複を取り除いたファイル数は，最低 1 件，最高 18 件，平均 2.3 件となった．その分布を図 14 に示す．図の表し方は図 12 と同様である．

基準 1 を満たさない 15 件のクエリについては，検索結果ごとに類似度の推定値を降順に並べたとすると，記録されているバージョンの順位は，2 位から 14 位，平均して 5.3 位であった．

基準 2 を満たしたクエリは 197 件中 187 件であった．残りの 10 件は，再利用元のバージョンとクエリとの類似度が真の値でも最高値をとらないものである．したがって，類似度に真の値ではなく推定値を用いたことによる影響の大きさは，基準 2 を満たす 187 件中のクエリのうち基準 2 を満たし基準 1 を満たさない 10 件，5.3%であると考えられる．

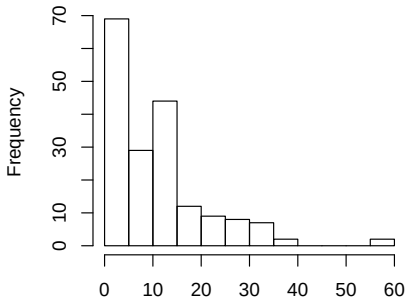

図 13: 各クエリの検索結果中で類似度の推定値が最高値をとるファイル数の分布

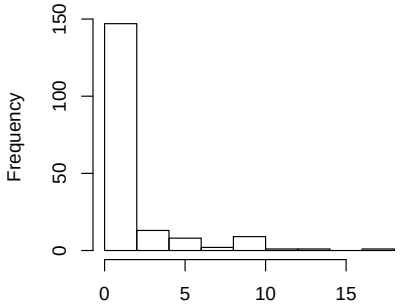

図 14: 各クエリの検索結果中で類似度の推定値が最高値をとるものの中で，コメント，フォーマットの違いを無視し重複を除いたファイル数の分布

表 5: 検索クエリとして用いるソースファイル

ID	ファイル名	ベクトルの一致数	検索結果の件数	LOC	サイズ [B]
0	mozpngconf.h	0	0	525	27,513
1	pngwrite.c	14,842	503	1,590	51,254
2	pngwtran.c	26,355	453	572	17,279
3	pngrtran.c	41,179	818	4,296	147,369

5.3 パフォーマンスの評価

提案手法のパフォーマンスについて評価した．実行環境の CPU は，Intel(R) Xeon(R) CPU E5-2620 が 2 プロセッサであり，メモリは 64GB，OS は Windows(R) 7 Professional である．また，データベースや検索クエリのファイルはすべて SSD に保存されている．

評価に使用するデータベースは，5.2 と同様である．

5.3.1 登録に対する評価

データベースは 200 プロジェクトから合計 567113 件のファイルを登録したものである．登録のために Git のリポジトリから一度すべてのファイルを取り出してディレクトリに保存し，その後データベースへの登録を行った．ディレクトリから取り出したファイルすべてを登録するために要した時間は 230 分であった．

5.3.2 検索に対する評価

検索のクエリとして，v8monkey のコミット 3a04be0690dff135ec42784557fedbf6c572cd22 の modules/libimg/png 以下に含まれている 4 ファイルを使用した．表 5 にそれらについて示す．検索に使用するベクトルのデータベース中で一致する数が多いほど実行時間がかかることが予想されるため，その数が固まらないように 5.2 で用いたクエリの中から選定した．また，ベクトルの一致数が 5.2 で用いたクエリの中で最大になるものを含めた．

検索を行った際にデータベースの内容の一部を OS がメモリにキャッシュするため，後続のクエリの速度が先行するクエリの検索の影響を受ける．このことを考慮し，4 つのファイルの検索の順序を全通り，つまり $4! = 24$ 通りの順序で検索を行い，その実行時間を測定した．24 通りのそれぞれの検索の間で OS のキャッシュをクリアし，同一ファイルの検索がキャッシュに影響されないようにした．

まず検索クエリとなるソースファイルをディスクから読み出し，MinHash のベクトルに変換するまでの時間を図 15 に示す．横軸は表 5 に示す ID を，縦軸は実行時間をミリ秒で表している．

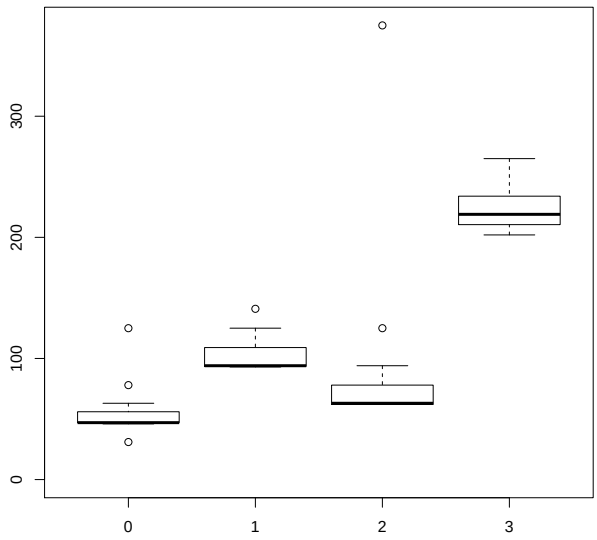

図 15: クエリとなるファイルの読み出しおよびベクトルへの変換に要した時間

次に，データベースからの検索にかかる時間を図 16 に示す．横軸は ID を，縦軸は実行時間をミリ秒で表している．ここで，ID は 4 で除した値が表 5 に示す ID を，剰余がそのクエリが何番目に検索が行われたかを示す．例えば，ID が 4 の倍数になっているものはキャッシュをクリアしてから最初に検索された場合の結果を示しており，剰余が 3 である場合は他の 3 クエリを検索した後に該当するクエリを検索した場合の結果を示している．

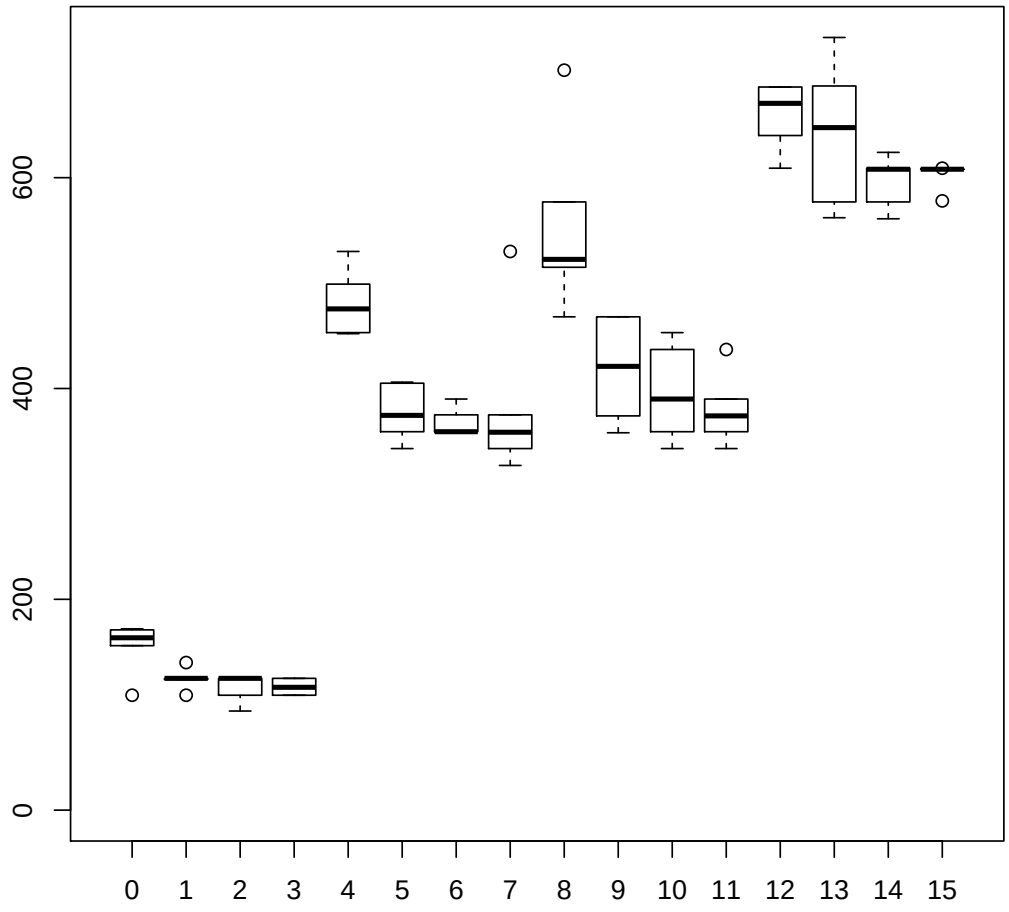

図 16: データベースからの検索の実行時間

6　妥当性への脅威

本研究の実装では 64bit のハッシュ長を用い，ハッシュ関数の振る舞いが完全にランダムにふるまうことを仮定し，その上でハッシュ値の衝突する確率が非常に低いとして衝突した場合の影響を考慮していない．しかしながら，実際に衝突が発生したのか，また衝突が発生していた場合の結果への影響についての検証をしていない．また，実装したハッシュ関数の振る舞いについての検証がなされていない．ハッシュ関数の偏りがケーススタディの結果に影響を与えた可能性がある．

ケーススタディで検索のクエリとして用いたプロジェクトは，1 プロジェクトのみである．このため，他のプロジェクトや，他の言語が利用されているプロジェクトに対し，この手法を適用した場合の有効性は本ケーススタディで確認した有効性とは異なる可能性がある．

5.2 で結果を分類するために設けた基準 2 において時間の制約上，類似度を求める対象を検索結果に表れたファイルのみに限定した．しかしながら，検索結果に表れなかったファイルの中に，クエリとの類似度が検索結果に表れたファイル以上になるものが存在する可能性がある．

7　まとめ

本研究では，検索のクエリとして与えられたソースコードのファイルに対し，大量のソースコードのファイルの中からソースコードの内容がクエリに類似するものを検索する手法を提案した．locality-sensitive hashing を用いることにより，高速な検索を実現した．また，提案手法を用いて実験を行い，実際に類似度の高いファイルが結果に表れることを確認した．ケーススタディでは，別コミットに含まれるファイルを合わせた延べ 197 ファイルを対象とし検索を行い，そのうちの全 197 件について，コミットメッセージに記録されていた再利用元を検索することができた．また，その中でも 182 件については記録されていた再利用元の類似度の推定値が，検索結果中の推定値の最高値となった．さらに，検索時間については 1 件につき 1 秒以内で検索を完了した．

本研究での提案手法は，検索のために 1 つのソースファイルを指定して検索を行う必要がある．今後の課題としては，プロジェクト全体のソースコードを与え，その中から再利用とそうでないものを判別することがあげられる．また，提案手法では 1 つのファイルについての情報しか利用していないが，入力として複数ファイルを与え，それらの情報を合わせて利用することで手法の精度の向上が期待できる．

謝辞

大阪大学大学院情報科学研究科コンピュータサイエンス専攻井上 克郎教授には，大変お世話になりました．大学生，大学院生としての 3 年間を実りあるものにできたこと，本研究を含めた研究活動を遂行できたのは，井上先生の御支援，御指導があってこそできたことです．心より深く感謝いたします．

大阪大学大学院情報科学研究科コンピュータサイエンス専攻松下 誠准教授には，特に発表練習の際に多くの，そして重要な質問やご意見をいただきました．それらによって発表および研究の不十分な点，改善点を直接的にも間接的にも知ることができました．心より深く感謝いたします．

大阪大学大学院情報科学研究科コンピュータサイエンス専攻石尾 隆助教には，直接の御指導をいただきました．本研究に限らず，研究の基礎となる部分から論文執筆，発表まで，多くのお力添えをいただきました．また，頻繁に助言をお願いすることもありましたが，その度に熱心に御指導いただき，本研究をここまで形にすることができました．心より深く感謝いたします．

大阪大学大学院情報科学研究科コンピュータサイエンス専攻神田 哲也氏には，研究中に発生した問題の原因を指摘していただいたり，論文の原稿にご意見をいただけたこと等によって，研究活動を円滑に進めることができました．心より深く感謝いたします．

最後に，その他様々な御指導，御助言等をいただきました大阪大学大学院情報科学研究科コンピュータサイエンス専攻井上研究室の皆様に心より深く感謝いたします．

参考文献

[1] Dumitru Brinza, Matthew Schultz, Glenn Tesler, and Vineet Bafna. Rapid detection of gene-gene interactions in genome-wide association studies. *Bioinformatics*, Vol. 26, No. 22, pp. 2856–2862, 2010.

[2] Andrei Z. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pp. 21–29, Jun 1997.

[3] Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pp. 271–280, New York, NY, USA, 2007. ACM.

[4] Julius Davies, Daniel M. German, Michael W. Godfrey, and Abram Hindle. Software bertillonage: Finding the provenance of an entity. In *Proceedings of the 8th Working Conference on Mining Software Repositories*, pp. 183–192, 2011.

[5] Julius Davies, Daniel M. German, Michael W. Godfrey, and Abram Hindle. Software bertillonage: Determining the provenance of software development artifacts. *Empirical Software Engineering*, Vol. 18, pp. 1195–1237, 2013.

[6] Lars Heinemann, Florian Deissenboeck, Mario Gleirscher, Benjamin Hummel, and Maximilian Irlbeck. On the extent and nature of software reuse in open source java projects. In *Proceedings of the 12th International Conference on Software Reuse*, Vol. 6727 of *Lecture Notes in Computer Science*, pp. 207–222, 2011.

[7] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pp. 604–613, New York, NY, USA, 1998. ACM.

[8] Katsuro Inoue, Yusuke Sasaki, Pei Xia, and Yuki Manabe. Where does this code come from and where does it go? – integrated code history tracker for open source systems –. In *Proceedings of the 34th International Conference on Software Engineering*, pp. 331–341, 2012.

[9] Lingxiao Jiang, Ghassan Misherghi, Zhendong Su, and Stephane Glondu. Deckard: Scalable and accurate tree-based detection of code clones. In *Proceedings of the 29th*

International Conference on Software Engineering, ICSE '07, pp. 96–105, Washington, DC, USA, 2007. IEEE Computer Society.

[10] Yushi Jing and Shumeet Baluja. Visualrank: Applying pagerank to large-scale image search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 30, No. 11, pp. 1877–1890, Nov 2008.

[11] Tetsuya Kanda, Takashi Ishio, and Katsuro Inoue. Extraction of product evolution tree from source code of product variants. In *Proceedings of the 17th International Software Product Line Conference*, pp. 141–150, Tokyo, Japan, 2013. ACM.

[12] Naohiro Kawamitsu, Takashi Ishio, Tetsuya Kanda, Raula Gaikovina Kula, Coen De Roover, and Katsuro Inoue. Identifying source code reuse across repositories using LCS-based source code similarity. In *Proceedings of the 14th International Working Conference on Source Code Analysis and Manipulation*, pp. 305–314, 2014.

[13] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. *Mining of Massive Datasets*, chapter 3. Cambridge University Press, 2014.

[14] Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. Detecting near-duplicates for web crawling. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pp. 141–150, New York, NY, USA, 2007. ACM.

[15] Parastoo Mohagheghi, Reidar Conradi, Ole M. Killi, and Henrik Schwarz. An empirical study of software reuse vs. defect-density and stability. In *Proceedings of the 26th International Conference on Software Engineering*, pp. 282–291, May 2004.

[16] Charikar Moses S. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pp. 380–388, New York, NY, USA, 2002. ACM.

[17] Jing Qiu, Xiaohong Su, and Peijun Ma. Library functions identification in binary code by using graph isomorphism testings. In *Proceedings of the 22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering*, pp. 261–270, 2015.

[18] Julia Rubin, Krzysztof Czarnecki, and Marsha Chechik. Managing cloned variants: A framework and experience. In *Proceedings of the 17th International Software Product Line Conference*, pp. 101–110, August 2013.

[19] Andreas Sæbjørnsen, Jeremiah Willcock, Thomas Panas, Daniel Quinlan, and Zhendong Su. Detecting code clones in binary executables. In *Proceedings of the 18th ACM International Symposium on Software Testing and Analysis*, pp. 117–128. ACM, 2009.

[20] Pei Xia, Makoto Matsushita, Norihiro Yoshida, and Katsuro Inoue. Studying reuse of out-dated third-party code in open source projects. *JSSST Computer Software*, Vol. 30, No. 4, pp. 98–104, 2013.

[21] 山中裕樹, 崔恩瀞, 吉田則裕, 井上克郎. 情報検索技術に基づく高速な関数クローン検出. 情報処理学会論文誌, Vol. 55, No. 10, pp. 2245–2255, oct 2014.

付録

5.2 および 5.3 のケーススタディで使用したプロジェクトの一覧を表 6 に示す.

表 6: ケーススタディで使用したプロジェクトの一覧

ID	プロジェクト名	リポジトリ URL
1	libgit2/libgit2	https://github.com/libgit2/libgit2.git
2	nmathewson/Libevent	https://github.com/nmathewson/Libevent.git
3	libuv/libuv	https://github.com/libuv/libuv.git
4	joyent/libuv	https://github.com/joyent/libuv.git
5	OpenKinect/libfreenect	https://github.com/OpenKinect/libfreenect.git
6	jedisct1/libsodium	https://github.com/jedisct1/libsodium.git
7	chaoslawful/ccard-lib	https://github.com/chaoslawful/ccard-lib.git
8	lukeredpath/libPusher	https://github.com/lukeredpath/libPusher.git
9	ladislav-zezula/StormLib	https://github.com/ladislav-zezula/StormLib.git
10	bither/bither-android-lib	https://github.com/bither/bither-android-lib.git
11	libopencm3/libopencm3	https://github.com/libopencm3/libopencm3.git
12	monome/libmonome	https://github.com/monome/libmonome.git
13	edenhill/librdkafka	https://github.com/edenhill/librdkafka.git
14	facebook/libphenom	https://github.com/facebook/libphenom.git
15	warmcat/libwebsockets	https://github.com/warmcat/libwebsockets.git
16	libtom/libtomcrypt	https://github.com/libtom/libtomcrypt.git
17	libarchive/libarchive	https://github.com/libarchive/libarchive.git
18	activesys/libcstl	https://github.com/activesys/libcstl.git
19	libimobiledevice/libimobiledevice	https://github.com/libimobiledevice/libimobiledevice.git
20	dinhviethoa/libetpan	https://github.com/dinhviethoa/libetpan.git
21	stephane/libmodbus	https://github.com/stephane/libmodbus.git
22	ellzey/libevhtp	https://github.com/ellzey/libevhtp.git
23	the-tcpdump-group/libpcap	https://github.com/the-tcpdump-group/libpcap.git
24	fukuchi/libqrencode	https://github.com/fukuchi/libqrencode.git
25	atgreen/libffi	https://github.com/atgreen/libffi.git
26	sustrik/libmill	https://github.com/sustrik/libmill.git
27	leaflabs/libmaple	https://github.com/leaflabs/libmaple.git
28	libharu/libharu	https://github.com/libharu/libharu.git
29	libav/libav	https://github.com/libav/libav.git
30	libusb/libusb	https://github.com/libusb/libusb.git
31	biomimetics/imageproc-lib	https://github.com/biomimetics/imageproc-lib.git
32	marforic/imagemagick_lib_iphone	https://github.com/marforic/imagemagick_lib_iphone.git
33	libass/libass	https://github.com/libass/libass.git
34	steve-m/librtlsdr	https://github.com/steve-m/librtlsdr.git
35	libMesh/libmesh	https://github.com/libMesh/libmesh.git
36	xxorde/librekinect	https://github.com/xxorde/librekinect.git
37	bashrc/libdeep	https://github.com/bashrc/libdeep.git
38	thom311/libnl	https://github.com/thom311/libnl.git
39	NetEase/libpomelo	https://github.com/NetEase/libpomelo.git
40	libhybris/libhybris	https://github.com/libhybris/libhybris.git
41	bitmovin/libdash	https://github.com/bitmovin/libdash.git
42	cisco/libsrtp	https://github.com/cisco/libsrtp.git
43	openSUSE/libsolv	https://github.com/openSUSE/libsolv.git
44	libgd/libgd	https://github.com/libgd/libgd.git
45	mongodb/libbson	https://github.com/mongodb/libbson.git
46	isislovecruft/library	https://github.com/isislovecruft/library.git

47	jcdutton/libbeauty	https://github.com/jcdutton/libbeauty.git
48	jcupitt/libvips	https://github.com/jcupitt/libvips.git
49	pkelsey/libuinet	https://github.com/pkelsey/libuinet.git
50	happyfish100/libfastcommon	https://github.com/happyfish100/libfastcommon.git
51	pbatard/libwdi	https://github.com/pbatard/libwdi.git
52	libvmi/libvmi	https://github.com/libvmi/libvmi.git
53	metajack/libstrophe	https://github.com/metajack/libstrophe.git
54	libimobiledevice/libplist	https://github.com/libimobiledevice/libplist.git
55	chokkan/liblbfgs	https://github.com/chokkan/liblbfgs.git
56	facebook/liblogfaf	https://github.com/facebook/liblogfaf.git
57	vincenthz/libjson	https://github.com/vincenthz/libjson.git
58	client9/libinjection	https://github.com/client9/libinjection.git
59	vstakhov/libucl	https://github.com/vstakhov/libucl.git
60	erikd/libsndfile	https://github.com/erikd/libsndfile.git
61	sam-github/libnet	https://github.com/sam-github/libnet.git
62	libtrading/libtrading	https://github.com/libtrading/libtrading.git
63	LibVNC/libvncserver	https://github.com/LibVNC/libvncserver.git
64	cjlin1/liblinear	https://github.com/cjlin1/liblinear.git
65	webmproject/libvpx	https://github.com/webmproject/libvpx.git
66	rampantpixels/foundation_lib	https://github.com/rampantpixels/foundation_lib.git
67	WaterJuice/CryptLib	https://github.com/WaterJuice/CryptLib.git
68	PocketInsanity/libSDL	https://github.com/PocketInsanity/libSDL.git
69	smibarber/libSOIL	https://github.com/smibarber/libSOIL.git
70	spark/core-common-lib	https://github.com/spark/core-common-lib.git
71	slavavdovichenko/MediaLibDemos	https://github.com/slavavdovichenko/MediaLibDemos.git
72	openwebos/nyx-lib	https://github.com/openwebos/nyx-lib.git
73	mysqludf/lib_mysqludf_json	https://github.com/mysqludf/lib_mysqludf_json.git
74	mysqludf/lib_mysqludf_sys	https://github.com/mysqludf/lib_mysqludf_sys.git
75	jeffminton/lib	https://github.com/jeffminton/lib.git
76	ArcticaProject/nx-libs	https://github.com/ArcticaProject/nx-libs.git
77	cloudwu/cstring	https://github.com/cloudwu/cstring.git
78	chewing/libchewing	https://github.com/chewing/libchewing.git
79	zhemao/libds	https://github.com/zhemao/libds.git
80	sahlberg/libiscsi	https://github.com/sahlberg/libiscsi.git
81	yixia/librtmp	https://github.com/yixia/librtmp.git
82	GomSpace/libcsp	https://github.com/GomSpace/libcsp.git
83	jedisct1/libpuzzle	https://github.com/jedisct1/libpuzzle.git
84	webmproject/libwebp	https://github.com/webmproject/libwebp.git
85	wolfcw/libfaketime	https://github.com/wolfcw/libfaketime.git
86	glennrp/libpng	https://github.com/glennrp/libpng.git
87	libreswan/libreswan	https://github.com/libreswan/libreswan.git
88	mono/libgdiplus	https://github.com/mono/libgdiplus.git
89	guardianproject/libsqlfs	https://github.com/guardianproject/libsqlfs.git
90	H2CO3/libsprec	https://github.com/H2CO3/libsprec.git
91	arnaudsj/libsvm	https://github.com/arnaudsj/libsvm.git
92	larskanis/libusb	https://github.com/larskanis/libusb.git
93	libimobiledevice/libirecovery	https://github.com/libimobiledevice/libirecovery.git
94	greatscottgadgets/libbtbb	https://github.com/greatscottgadgets/libbtbb.git
95	Chronic-Dev/libirecovery	https://github.com/Chronic-Dev/libirecovery.git
96	librsync/librsync	https://github.com/librsync/librsync.git
97	libtom/libtommath	https://github.com/libtom/libtommath.git
98	libguestfs/libguestfs	https://github.com/libguestfs/libguestfs.git

99	rescrv/libmacaroons	https://github.com/rescrv/libmacaroons.git
100	menudoproblema/libemqtt	https://github.com/menudoproblema/libemqtt.git
101	maxmind/libmaxminddb	https://github.com/maxmind/libmaxminddb.git
102	ktossell/libuvc	https://github.com/ktossell/libuvc.git
103	akumrao/libxcb	https://github.com/akumrao/libxcb.git
104	hackedteam/libpemelter	https://github.com/hackedteam/libpemelter.git
105	payden/libwebsock	https://github.com/payden/libwebsock.git
106	siddontang/libtnet	https://github.com/siddontang/libtnet.git
107	taf2/libebb	https://github.com/taf2/libebb.git
108	libimobiledevice/libusbmuxd	https://github.com/libimobiledevice/libusbmuxd.git
109	anholt/libepoxy	https://github.com/anholt/libepoxy.git
110	armon/libart	https://github.com/armon/libart.git
111	seanmiddleditch/libtelnet	https://github.com/seanmiddleditch/libtelnet.git
112	celery/librabbitmq	https://github.com/celery/librabbitmq.git
113	redjack/libcork	https://github.com/redjack/libcork.git
114	sahlberg/libnfs	https://github.com/sahlberg/libnfs.git
115	ClusterLabs/libqb	https://github.com/ClusterLabs/libqb.git
116	nickhutchinson/libdispatch	https://github.com/nickhutchinson/libdispatch.git
117	swift-nav/libswiftnav	https://github.com/swift-nav/libswiftnav.git
118	libvirt/libvirt	https://github.com/libvirt/libvirt.git
119	koanlogic/libu	https://github.com/koanlogic/libu.git
120	libfuse/libfuse	https://github.com/libfuse/libfuse.git
121	dparrish/libcli	https://github.com/dparrish/libcli.git
122	wiiudev/libwiiu	https://github.com/wiiudev/libwiiu.git
123	LudovicRousseau/libusbx	https://github.com/LudovicRousseau/libusbx.git
124	dropbox/librsync	https://github.com/dropbox/librsync.git
125	fbuihuu/libtree	https://github.com/fbuihuu/libtree.git
126	sharow/libconcurrent	https://github.com/sharow/libconcurrent.git
127	nfc-tools/libnfc	https://github.com/nfc-tools/libnfc.git
128	zedshaw/liblcthw	https://github.com/zedshaw/liblcthw.git
129	Theano/libgpuarray	https://github.com/Theano/libgpuarray.git
130	andlabs/libui	https://github.com/andlabs/libui.git
131	ofiwg/libfabric	https://github.com/ofiwg/libfabric.git
132	andrewrk/libsoundio	https://github.com/andrewrk/libsoundio.git
133	djmuhlestein/fx2lib	https://github.com/djmuhlestein/fx2lib.git
134	stoni/ta-lib	https://github.com/stoni/ta-lib.git
135	liyuming1978/NativeLibCompression	https://github.com/liyuming1978/NativeLibCompression.git
136	senojsitruc/libEmailz	https://github.com/senojsitruc/libEmailz.git
137	baz/libORCDiscount	https://github.com/baz/libORCDiscount.git
138	Starlon/LibVisual	https://github.com/Starlon/LibVisual.git
139	mirror/libX11	https://github.com/mirror/libX11.git
140	litl/lufa-lib	https://github.com/litl/lufa-lib.git
141	D-Programming-Deimos/libX11	https://github.com/D-Programming-Deimos/libX11.git
142	theck01/offbrand_lib	https://github.com/theck01/offbrand_lib.git
143	BrianSharpe/GPU-Noise-Lib	https://github.com/BrianSharpe/GPU-Noise-Lib.git
144	ynezz/librs232	https://github.com/ynezz/librs232.git
145	OpenSC/libp11	https://github.com/OpenSC/libp11.git
146	gphoto/libgphoto2	https://github.com/gphoto/libgphoto2.git
147	GNOME/libxml2	https://github.com/GNOME/libxml2.git
148	librepilot/LibrePilot	https://github.com/librepilot/LibrePilot.git
149	NetEase/libpomelo2	https://github.com/NetEase/libpomelo2.git
150	memononen/libtess2	https://github.com/memononen/libtess2.git

151	bji/libs3	https://github.com/bji/libs3.git
152	libssh2/libssh2	https://github.com/libssh2/libssh2.git
153	tmzt/androix-lib-libX11	https://github.com/tmzt/androix-lib-libX11.git
154	JediKnight/lib	https://github.com/JediKnight/lib.git
155	jmadhav/lib	https://github.com/jmadhav/lib.git
156	yangjin-unique/lib	https://github.com/yangjin-unique/lib.git
157	clmrosey/LIB	https://github.com/clmrosey/LIB.git
158	humar/lib	https://github.com/humar/lib.git
159	greg198584/lib	https://github.com/greg198584/lib.git
160	saumurf/lib	https://github.com/saumurf/lib.git
161	mdwarfgeek/lib	https://github.com/mdwarfgeek/lib.git
162	rambo/TinyWire	https://github.com/rambo/TinyWire.git
163	zhengshuxin/acl	https://github.com/zhengshuxin/acl.git
164	brainfucker/hashlib	https://github.com/brainfucker/hashlib.git
165	gulyan/lib	https://github.com/gulyan/lib.git
166	xuefeng529/lib	https://github.com/xuefeng529/lib.git
167	spolitov/lib	https://github.com/spolitov/lib.git
168	brucewmj/LIB	https://github.com/brucewmj/LIB.git
169	SteelTermite/lib	https://github.com/SteelTermite/lib.git
170	b-andris/lib	https://github.com/b-andris/lib.git
171	alex8092/Lib	https://github.com/alex8092/Lib.git
172	cetorchia/lib	https://github.com/cetorchia/lib.git
173	linflove/lib	https://github.com/linflove/lib.git
174	tracymacding/lib	https://github.com/tracymacding/lib.git
175	ihzr/lib	https://github.com/ihzr/lib.git
176	devendradora/lib	https://github.com/devendradora/lib.git
177	maksspace/lib	https://github.com/maksspace/lib.git
178	aroemer/lib	https://github.com/aroemer/lib.git
179	cglens/lib	https://github.com/cglens/lib.git
180	JugglerShu/lib	https://github.com/JugglerShu/lib.git
181	umbs/Lib	https://github.com/umbs/Lib.git
182	rohanmaddamsetti/lib	https://github.com/rohanmaddamsetti/lib.git
183	bhagyashrikshirsagar1/lib	https://github.com/bhagyashrikshirsagar1/lib.git
184	ggila/lib	https://github.com/ggila/lib.git
185	marquezc/lib	https://github.com/marquezc/lib.git
186	theryee/lib	https://github.com/theryee/lib.git
187	kuvo1017/lib	https://github.com/kuvo1017/lib.git
188	sakeven/lib	https://github.com/sakeven/lib.git
189	TestKitMaster/Lib	https://github.com/TestKitMaster/Lib.git
190	pplinlin3/lib	https://github.com/pplinlin3/lib.git
191	dulm-today/lib	https://github.com/dulm-today/lib.git
192	dulrich/lib	https://github.com/dulrich/lib.git
193	gaccob/gbase	https://github.com/gaccob/gbase.git
194	trondn/libmemcached	https://github.com/trondn/libmemcached.git
195	membase/libvbucket	https://github.com/membase/libvbucket.git
196	jons/libnmea	https://github.com/jons/libnmea.git
197	StarchLinux/libxcb	https://github.com/StarchLinux/libxcb.git
198	toymachine/libredis	https://github.com/toymachine/libredis.git
199	dajobe/librdf	https://github.com/dajobe/librdf.git
200	kralf/libipc	https://github.com/kralf/libipc.git