Webサイト上のPythonソースコードと説明文の自動対応付けによるソースコード検索

柴 友行　　酒井 浩之　　脊戸 和寿
成蹊大学 理工学部 情報科学科
us152059@cc.seikei.ac.jp, {h-sakai,seto}@st.seikei.ac.jp

1 はじめに

インターネットのWebサイト上にプログラムソースコードは多く存在しており，自由に閲覧ができる．プログラミング学習において自分でソースコードを記述する際，インターネット上のプログラムソースコードを参考にして学習する場面は多くある．多くの人が学生時代，プログラミングの課題をgoogle検索を用いて似たようなソースコードがないか探した経験はあるのではないだろうか．しかし検索結果の上位のWebサイトに必ずしも望んだプログラムがあるとは限らない．なぜならば，google検索など従来の検索エンジンではプログラムのソースコードの直接的な検索は現状不可能なためである．直接的な検索結果としてソースコードが出力されず，目的のソースコードを得るためには検索結果のページ群を探索する必要がある．学習したいアルゴリズム名などで検索し，検索したアルゴリズムに対応したソースコードを表示できるようになれば，プログラミング学習において検索結果から参考にしたいソースコードを探索する手間が省けるので，学習の効率化が可能になる．そこで本研究ではプログラミング言語としてPythonを対象とし，検索要求のアルゴリズム名や，動作などの説明文と収集したWebサイト内のソースコード説明文との対応を取り，ソースコードを検索する手法の開発を行った．

関連研究としてTF-IDF法を用いて一対のソースコード同士の対応を取る手法や，統計的機械翻訳を用いて一行の自然言語からの一行のソースコードを生成する研究が現在までに行われている [1], [2]．前者の研究では本研究と同様，プログラミング言語で用いられる予約語をソースコードの同士の対応付けに利用し非常に高い精度でソースコード同士の対応付けに成功しているが，アルゴリズム名や動作文での検索はできない．後者の研究ではソースコードの出力はできるもののプログラミング構文との相違点が存在したり，複数行の実用的なソースコードの出力ができないといった問題点が存在する．また数学的記事を取り扱うwikipediaのページから数式とその名前，説明文を抽出する研究も存在する [3]．この研究ではmathタグの中が数式であるという法則性があるが，本研究はある特定のタグにソースコードが記述されているといった法則性が存在しないためソースコードのそうでない英文とを判別する手法から考案する必要がある．本研究ではインターネット上のPythonのプログラミングWebサイトからソースコードとその説明文の対応を獲得，アルゴリズム名などの固有名詞と簡単な処理内容文でのソースコードの検索を行う手法を提案する．

2 Pythonプログラムソースコードの抽出

本章では，インターネット上のWebサイト (約24GB分) からPythonソースコードを抽出する手法について説明する.Webページ上からソースコードを抽出するイメージを図1に示す．

2.1 ソースコードの判別

プログラムのソースコードには，インデントに使用するタブ文字，スペースや改行，括弧やカンマなどの特殊記号 (HTML内で利用されている記号も例外ではない)，プログラム言語側から既に指定されている予約語が多く

含まれる．そのため，HTML 内の BODY タグに直接記述してしまうと HTML 本体の記述形式と重複し，Web ページそのものや記述したソースコードが意図しない表示をしてしまう可能性が非常に高い．しかし HTML 側から用意されている整形済みテキストタグ (以下 pre タグ) 内に記述することで，ソースコードを記述したそのままの形式で Web ページに表示することが可能になる．そのため，ソースコードを抽出する際には HTML 全文を検索するより pre タグ内の文字列のみに絞った方が効率的に実行できる．しかし pre タグ内には通常の英文なども記述される場合があるため，それらとソースコードを判別しなければならない．本手法ではソースコードを判別するためにプログラム言語で予め指定されている予約語を文字列内から検索し，その含有率が大きければソースコードであると判断する．以下に通常の英文とソースコードでの予約語含有率例を示す．pre タグ内文字列の分析にはオープンソースソフトウェアの nltk を利用した．

- ソースコードの判別例

```
def foo ( x ) :
    if x >= 0 :
        return math. sqrt ( x )
    else :
        return None
def bar ( x ) :
    if x < 0 :
        return None
    return math. sqrt ( x )
```

含有率：9/39 ≒ 0.23

上記の例ではソースコードの予約語含有率が通常の英文の予約語含有率を上回っている．

ソースコード内には，プログラミング言語の仕様上特別な意味を持つ語が存在する． これを予約語と呼ぶ.(Python においては，and，break，continue，else，for，if，print 等. 全 31 語.) 文字列内の予約語含有率 ($reservedRate(words)$) を以下のように定義する．

$$reservedRate(words) = \frac{\sum_{i=0}^{|words|} reserved(words(i))}{|words|} \quad (1)$$

- words : pre タグ内の文字列に含まれる単語の総集合
- reserved : 与えられた単語が予約語なら 1，そうでなければ 0 を返す関数

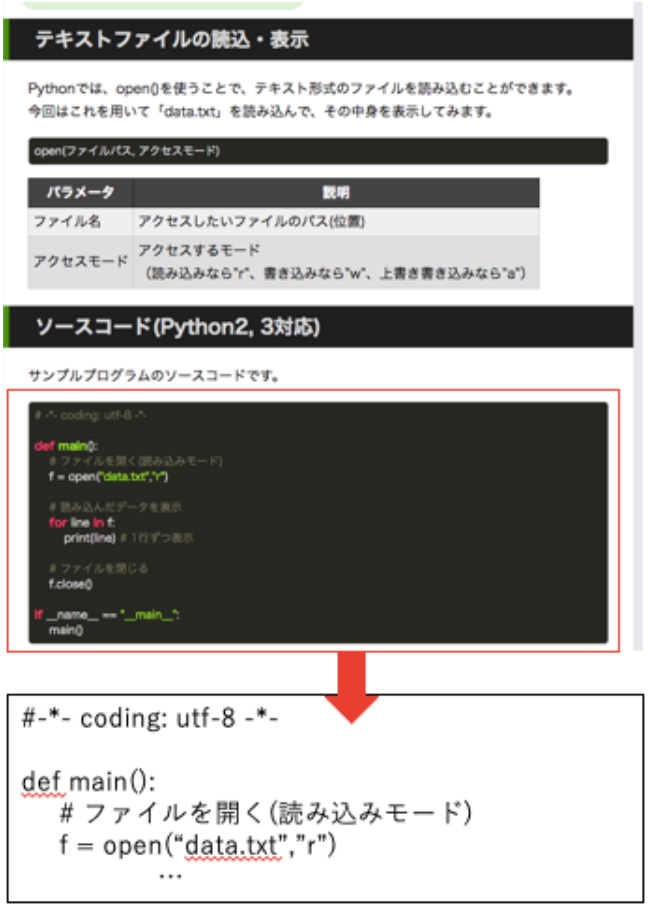

図 1: ソースコードの抽出例

プログラムソースコード内の全ての単語が予約語であるわけでは無いが，一定の割合は必ず含まれている．そこで予約語含有率が一定の閾値以上を示した pre タグ内文字列をソースコードと判断する．本研究では閾値を「0.1 以上」に設定したところ，ソースコードが最も多く抽出できた．また人手にて判別した結果と比較し，約 8 割の精度を得た．以下に条件分岐に関して記述された Web サイトから抽出できたソースコードの例を示す．

```
def if_test(num):
    if num > 100:
        print('100 < num')
    elif num > 50:
        print('50 < num <= 100')
    elif num > 0:
        print('0 < num <= 50')
    elif num == 0:
        print('num == 0')
```

3 説明文記述部分の選定とソースコードとの対応付け

プログラムの説明文と対応するソースコードは同じ Web ページ上に記述されていることが殆どである. 前章で提案した手法を用いてソースコードと判断した pre タグ内文字列が記述されている HTML 内の p タグを説明文として対応付けた. 以下に抽出出来た説明文とソースコードの例を示す. ※なお，深層学習による説明文の抽出を試みたが、大量の学習データの作成が困難でありうまくいかなかった.

・説明文
繰り返しやループですね.
・ソースコード

```
for var in ['Python', 'javascript', 'Rudy']:
    print(var)
```

4 ソースコードの検索

本節では，前節までに説明した手法で構成されたプログラムソースコード検索システムについて記述する. Python のプログラム Web サイトを検索した結果から無作為に HTML を 24GB 分収集したものを検索対象データ群とした.

1. 検索文を入力，形態素解析して TF-IDF 値を要素値とした文ベクトルを形成
2. 全検索対象ページを走査してプログラムソースコードが存在するページ内の全 p タグ内の平文をベクトル化
3. 各ベクトルと検索文ベクトルとの Cosin 距離を計算して値が 0 以上ならば Cosin 距離，ソースコード，p タグを一つのオブジェクトとして保存
4. 全ページを走査後 4 で保存したオブジェクトを Cosin 距離順に並び替え，上位のオブジェクトを表示

5 評価

作成した検索システムを用いてアルゴリズム名を検索語，もしくはプログラム動作を説明する文を検索文として複数検索し，本手法によりそれぞれ対応するソースコードが上位 n 件以内に出るか判定する. また本手法の有効性を確立するため，ベースラインとして本研究で評価したものと同じ検索語や検索文と内部の平文がフレーズ一致した HTML を出力，目的のソースコードが出力出来るかも判定した. アルゴリズム名での検索結果では，全 15 件中 12 件検索語に対応するソースコードが出力されている. 対して，プログラム動作を説明する検索文での検索結果では， 全 11 件中 5 件検索文に対応するソースコードが出力されている. またベースラインに関してはアルゴリズム名での検索では全 15 件中 3 件，プログラム動作を説明する検索文では全 11 件中 1 件対応するソースコードが出力されている. 表 1，表 2 より本手法の精度がベースラインの精度を大きく上回っていることが分かり，本手法が有効であることを示すことができた. 以下に実装した検索システムでの検索例を示す. 表 1，表 2 に結果の一部を示す.

検索文：「リストの最大値を求める」

最大値を探すアルゴリズムとほとんど同じです.

```
==========================================
math_max_student = taro
for student in [taro, jiro, saburo]:
   if(math_max_student.math < student.math):
        math_max_student = student

print(math_max_student.name)
# jiro
```

表 1: アルゴリズム名での検索評価結果 (一部)

検索語	n=1	n=3	n=5	ベースライン
クイックソート	○	／	／	○
二分木	○	／	／	×
最大値	○	／	／	×
ヒープソート	×	ー	ー	○
二分探索	○	／	／	×
双方向リスト	○	／	／	×

表 2: プログラム動作を説明する文での検索評価結果 (一部)

検索文	n=1	n=3	n=5	ベースライン
リストの最大値を求める	○	／	／	×
リストの合計を求める	○	／	／	×
値の足し算をする	○	／	／	×
条件分岐する	○	／	／	×
素数を求める	○	／	／	○

6 考察

ある特定のアルゴリズムであれば本研究の手法を用いて検索しソースコードとの関連付けが可能であることが判明した. その過程において，予約語をある程度含んでいればソースコードである確率が高いことを示した. しかし検索結果の説明文やソースコードが想定されるものと一致しなかった検索語，検索結果そのものが出力されなかった検索語が存在した. 例えば，前者は以下のような場合である.

検索文 :「根号の計算をする」
検索結果 :
2 つの計算もできるのです.
-> 0.9342122497731796
==========
print((1+2+3)*4-5)

rate : 0.14285714285714285

この検索結果の説明文は「計算」という名詞が含まれているために上位に出力されたと考えられる. 検索結果の説明文に誤った文が挙がったため，追随して検索結果のソースコードも誤ったものが挙がってしまった. 本手法では名詞のみで関連度を計算しているため，同じ名詞を用いた文同士であれば関連度が高くなってしまう.
後者の原因として考えられるのは，収集した検索対象データ群内に対象のソースコードが存在しなかった，Python のライブラリとして既に実装されている為実装の必要性がなく Web 上に掲載されていない，対象のソースコードが検索対象の HTML タグ内に記述されていなかった，などである. 検索対象のデータ範囲を抽出した Web 上の HTML だけでなくインターネット上の検索エンジンと併用したり，検索対象の HTML タグを pre タグのみではなく他の HTML タグも含めることで検索精度が向上する可能性がある. しかし検索対象のデータ範囲や検索対象の HTML タグ範囲を拡大すると検索にかかるコストも増大する為，実用的な検索システムに落としこむために検索対象の絞り込みなど手法の改善が必要になる.

7 まとめ

本稿では Python を対象として予約語を用いたソースコードの抽出とそれに対応する説明文の抽出，および対応付けによる検索システムの構築を行った. ソースコードの抽出において，予約語の含有率を計算することによりソースコードとそれ以外の文との判別を行った. 評価の結果，ソースコードの判別を行わず，単純なフレーズ一致を比較対象とした場合に比べて大幅に精度が向上した.

参考文献

[1] 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, 2014.

[2] 札場 寛之, 小田 悠介, Graham Neubig, 吉野 幸一郎, 中村 哲: 統計的機械翻訳を用いた自然言語からのソースコード生成. 言語処理学会第 22 回年次大会発表論文集, 2016.

[3] Minh Nghiem Quoc, Keisuke Yokoi , Yuichiroh Matsubayashi, and Akiko Aizawa: Mining coreference relations between formulas and text using Wikipedia. Proceedings of Second Workshop on NLP Challenges in the Information Explosion Era (NLPIX 2010), pages 69-74, Beijing, August 2010