Vol.2018-CE-143 No.25
2018/2/18

プログラム学習における説明文の使用語彙に関わる調査

竹内和広[†1]　間嶋義喜[†2]

概要：プログラムの教育においては、学習者の理解度合いに応じた説明の工夫が必要であると考えられる。本稿では、プログラムの入門書や解説の文書表現をデータに、それらの書き手がどのような表現をプログラムの学習者に親和性が高いと考えているか、その語用について調査する。具体的には、このようなプログラム記述の言語使用を、テキストマイニングの 1 手法である関連ネットワークを、調査対象とする文章における語の共起に基づいて作成する。そのような複数のテキストに対して作成した関連ネットワークの組織比較・対照を行い、プログラム理解に関わる概念がどのような自然言語表現を用いて表現され、高度化されるかについて検討を行った。

キーワード：プログラム作成知識，概念表現，テキストマイニング，共起ネットワーク

An investigation of the technical vocabulary in explanatory documents in program learning

KAZUHIRO TAKEUCHI[†1]　YOSHIKI MASHIMA[†2]

Abstract: The use of natural language in explaining ideas of programming such as algorithms might be different from those of daily language. In technical understanding of programs, it is thought that it is necessary to devise explanation according to the understanding degree of learners. In this paper, we investigate the use of words in a textbook for the programming language and in explanatory documents for software algorithms. The purpose of the investigation is ton know what kind of vocabulary is considered as affinity for learners of the program is high. Specifically, we will construct word co-occurrence network that is one of the methods in text mining based on co-occurrence of words in those textbook or documents. Based on the network, we compare the language uses created from such documents and examine what kinds of natural language words are used to express concepts related to program understanding.

Keywords: Programming Knowledge, Conceptual Expression, Text Mining, Word Co-occurrence Network

1. はじめに

プログラム作成は情報技術における専門技能として重要な位置づけにある。プログラム作成の前提には、対象となる問題を解くための手順を定式化した形で表現する必要がある。このような手続きはアルゴリズムと呼ばれ、コンピュータが情報を処理するための基礎となる。

アルゴリズムには様々なレベルがあると同時に、それを説明するための様々な記法がある。例えば、自然言語、擬似コード、フローチャート、プログラミング言語といった記法である。そのような記法にはそれぞれに特徴がある。例えば、プログラミング言語は、自然言語のような曖昧性を持たないため､明確にアルゴリズムを記述できるものの、プログラミング言語を習得する必要があり、さらに、その実践的な記述内容を理解することは、経験の乏しいプログラム初学者には理解が困難であるという問題がある。

それに対して、フローチャートや擬似コードは、工夫をすれば、プログラム言語の習得よりも、直感的にアルゴリズムの構造を説明することができる。さらに自然言語の場合は、フローチャートや疑似コードとは異なり、誰もが日常的に用いているものであり、アルゴリズムを学ぶ事前学習の必要性から考えると、プログラムを体験していないプログラム初学者と教授者との間の共通言語としての役割として重要と言える。

他方、自然言語のみで複雑なアルゴリズムを説明することは困難である。そのため、多くの教科書や解説書、そして独立行政法人情報処理推進機構（IPA）が、合格対象者像を「高度 IT 人材となるために必要な基本的知識・技能をもち、実践的な活用能力を身に付けた者」と位置付ける基本情報技術者試験の問題でも、アルゴリズムを擬似コードやフローチャート、処理内容を示す図等の記法と併用して自然言語による説明を行うことが一般的である。また、自然言語は、プログラミング言語で記述されたプログラム中にも使用される。例えば、プログラム中の部分要素がどのような働きを持つかを説明するコメント文、変数や関数名などの識別子等の命名に自然言語が用いられる。このように、プログラミング言語で記述されるアルゴリズムは、たとえプログラミング言語を学習済であったとしても、自然

†1 大阪電気通信大学情報通信工学部
Osaka Electro-Communication University
†2 大阪電気通信大学大学院
Osaka Electro-Communication University

言語を介在せずにその機能や処理について説明することが困難であることを示している。

このように、専門的な技術であるプログラム技術、特にアルゴリズムを説明する上での自然言語は、日常的な言語使用とは異なる語用を持ち、そのような言語表現に不慣れなプログラム初学者は戸惑いを覚える可能性がある。例えば、プログラムに対する説明文書が煩雑で長く、通常の言語使用とは異なる意味をもっているために、その読み手が文章の意味について、理解が困難である可能性がある。

2. プログラミング言語の教科書に対する調査

プログラム作成を学ぶために、実際にプログラム実行をコンピュータで試すことができるプログラミング言語を学ぶことは、実践的な学習の導入となる。本稿では、プログラミング言語の入門的教科書として、数々の大学で採用されていることで知られる教科書[1]を題材として調査を行う。

この教科書の出版社からの書籍紹介としては、「経験がなく、はじめてプログラミングをはじめる人にも、無理なくプログラミングの基本から学習してもらえる」ように、次の工夫がなされているとされる。また、第五版は、全国学校図書館協議会選定図書となっている。

- 読みやすい解説でスラスラ読み進められる。
- 豊富なイラスト図解で、概念をイメージでわかる。
- たくさんのサンプルプログラムで、試して理解できる。

これらの特徴を客観的に見るために、同書のまえがき、目次、索引、プログラムリスト、図といった部分を除く、本文の自然言語による文章記述をコンピュータに入力し、出現語の共起ネットワークを作成したものを図 1 に示す。

共起ネットワークは、テキストマイニングの典型的な分析手法の一つである。なお、この作業はテキストマイニングのオープンソフトウェアである KH Coder[2]を用いて行った。KH Coder は自由記述アンケートの分析[3]等に柔軟に対応できるため、広く用いられているツールである。

テキストマイニングは、日本語に対する自然言語処理を共通前処理として行う。具体的には、日本語の文を形態素に分割する、形態素解析と呼ばれる処理を行う。形態素とは、意味を持つ最小単位である。直感的に言えば、意味を持つ最小単位は語であると考えられるが、例えば大阪電気通信大学という語は、大阪、電気、通信、大学という語により構成されているとも考え得る。このような語の認定、例えば大阪電気通信大学を固有名詞として認定するのか、固有名詞を構成している語を認定したいとするのかの要求は、分析者に依存して定義される度合いが高い。そのため、どのような分析要求であっても、広く対応できる単位として形態素という単位が導入されている。本稿では、形態素解析モジュールの MeCab[4]の解析結果として数字、数式、特殊記号の排除処理を行うが、形態素の結合する複合語処理については行わない。

なお、共起ネットワークに示された、語が形態素という言語単位であることは、前述した通りであるが、ネットワークのノードとして、どの語を選択するかも一定の基準があるわけではなく、分析者の分析主眼によって選択する必要がある。図 1 の語の選択は、教科書の自然言語記述に使われた内容語になりうる語の上位出現数を基準に出現回数 11 回以上の語をノードの対象として形式的に選択した。

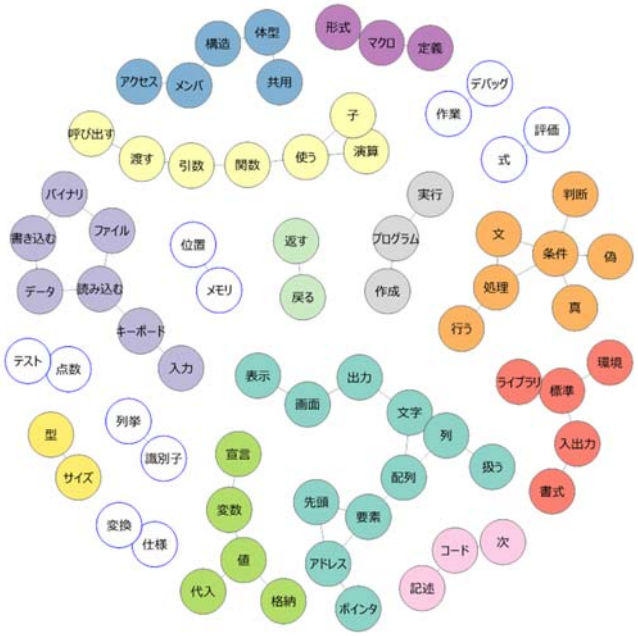

図 1 プログラミング言語の教科書から作成した共起ネットワーク

Figure 1 Word co-occurrence network constructed from a textbook for programming language

共起ネットワークは、語の概念を表現する関連語ネットワークの一種である。関連語ネットワークは、特定の語の概念を、他の語との関係により表現する手法である。語間の関係は、特定の文や段落、あるいは文章といった言語単位に同時に生起するかを基準に計量することが多い。そのような共起の計量手法には、Jaccard 係数、Dice 係数、Simpson 係数といったものが存在する。図 1 では、Jaccard 係数が 0.1 以上であるものを関係性ありとし、ノード間のアークとしてネットワークに表現している。

図 1 を見ると、プログラムの基本構造である、条件分岐や繰り返しだけではなく、C 言語というプログラミング言語の特徴である、配列、変数、ファイル入出力、構造体といった概念を良く表現できている。

もちろん、このような概念表現の記述力は限定的である。例えば、図 1 のネットワークで、「繰り返し」に関連付けがなされている「文」は、実際の教科書は、次のように記述している。

> *“C 言語では、1 つの小さな処理（「仕事」）の単位を文（statement）と呼び、最後に；（セミコロン）という記号をつけます。そして、この「文」が、原則として先頭から順番に、1 文ずつ処理されるということになっています。(p. 18)”*

上記の定義は、記憶装置にプログラムを内蔵し、それを先頭から順番に「１つずつ」逐次的に処理するノイマン型コンピュータの概念的な本質を分かりやすく説明しているが、そのような詳細にわたる概念は共起ネットワークには十全に表現されていない。しかし、調査対象とする文章が伝えようとする概念を、語の関係付けにより端的に視覚化できる点が、共起ネットワークによる概念表現の利点であるとも言える。

また、共起ネットワークによる概念の表現は、文章分析に対する客観的な基準により設定可能であることが特徴である。そのような基準の中で最も重要な基準は、ノードとして採用する語の選択と､既に述べた語間の類似度(距離)の定義（関係を持つ限界の閾値の設定を含む）である。前者の基準には、形態素解析の際に得られる品詞情報によって対象を限定することと、出現数によって限定するという典型的な手法がある｡すなわち､特定の文章で説明される、プログラム学習で習得すべき概念を、品詞情報による限定と、相対的な出現数による限定と二つの方向性から絞り込まれた語集合内の関係性によって表現できる。

図 1 に示したような、プログラミング言語の教科書で説明される、プログラムの基本構造や基本概念そのものは、専門家でなくとも体験的に理解できるとされる。例えば、GUI プログラミング言語である Scratch[5]では、情報を専門としない中学生や高校生でも条件分岐や繰り返しといったプログラムの基本構造の学習に効果的とされている。しかしながら､プログラミング言語の基礎概念だけの習得で、プログラム作成が十分に行えるわけではない。高度なプログラム作成には、習得した基礎概念の基盤の上に、さらに高度なプログラム作成の専門的概念の習得が段階的に必要であるものと考えられるが、図 1 に表現されたような基礎概念が、どのような形で高度化されるかについては明らかではない。

3. リポジトリからのメソッド語抽出

前節ではプログラム初学者が使う教科書を用いて、プログラム作成がどのように導入されるかを、テキストマイニングの 1 手法である関連ネットワークによって表現することを試みた。ここでは、前節とは対照的に、実際に動作するプログラムを記述している経験を積んだプログラム作成者が、プログラムの部分要素が果たす機能をどのような言語表現によって使用するかを調査する。具体的には、プログラミング言語によるプログラム記述中に用いられる語に着眼した調査を行う。

本節の調査対象とするプログラム例はプログラミング言語 Java（以降、単に Java と記す）で記述されたものとした。Java は、サン・マイクロシステムズにより、C++/C 言語の代替となるプログラミング言語として開発された[6]。Java の特徴は様々なものがあるが、オブジェクト指向を採用していることがあげられる。Java では、C 言語にオブジェクト指向を取り入れて拡張した C++言語にも類似し、C 言語の関数に相当する部分は、クラスに対してメソッドとして定義する機構になっている。

C 言語の関数の命名に関しては、前述した教科書でも、「関数のしくみを知る」として、毎月自分の貯金からお金を引き出す場合の「預金を引き出す」処理を、表 1 のような処理として例示している（同書 p.216)。

そして、このような処理のまとまりに、例えば「引き出し処理」という自然言語の名前をつけ、関数により構造的にプログラムを記述することを説明している。具体的な、説明は以下の通りである。

> *“C 言語でも、複雑なコードを書くようになってくると、たびたび行わなければならない一定の処理が出てくる場合があります。このような処理を、そのたびに何度も記述していくのはとても面倒です。そこでC 言語では、一定の処理をまとめて記述する関数（function）という機能を用意しています。(同書 p. 216)”*

表 1 関数にまとめられる処理列の例
Table 1 Example of organized statements for a function

順番	手続き
1	通帳を自動現金支払機に入れる
2	暗証番号を入力する
3	金額を指定する
4	お金を受けとる
5	お金と通帳を確認する

C 言語で関数として捉えられていた処理のまとまりは、オブジェクト指向プログラミング言語ではさらに精密化された機構に取り入れられている。オブジェクト指向のモデル化では、処理の対象となる文字列や数値といった抽象物をオブジェクトととらえ、その抽象物を処理するメソッドを定義する。例えば、整数をオブジェクトとするなら、整

数というオブジェクトの値を変えたり、値に他の整数値との加減乗除の演算を加えたりする処理をメソッドと呼ぶ。

このようなオブジェクトとメソッドの関係は、名詞に対する動詞の関係と密接な関係を持つ。例えば、ポチという犬クラスのオブジェクトに、「ポチを走らせる」、「ポチに食べ物を食べさせる」という処理を行わせることを *run(ポチ)*, *eat(ポチ, 食べ物)*と記述する、自然言語の述語項構造の記述特性をより反映した記述が求められる。

表 2 クラス定義に出現する上位メソッド語
Table 2 Top thirty method names occurred in class definitions

出現順位	メソッド語	定義例数
1	run	1173
2	toString	1148
3	main	1111
4	equals	700
5	hashCode	560
6	getName	541
7	actionPerformed	531
8	setUp	502
9	get	447
10	foo	440
11	add	387
12	execute	381
12	init	381
14	Line	341
15	apply	337
16	close	317
17	remove	282
18	Test	277
19	onClick	267
20	getId	248
21	update	240
22	getValue	238
23	getType	234
24	tearDown	229
25	read	219
26	process	218
27	create	214
28	write	209
29	configure	204
30	accept	199

本稿では、以上のような背景から、Java で記述された数理プログラムの実装において、処理を定義するメソッド名に用いられる語を調査することを考えた。すなわち、Java のプログラム記述において識別子の記述が C 言語に比べて、上記のようなオブジェクト指向の背景から慣例的に統制されていると考えたからである。

調査対象は、オープンソースで公開されているソフトウェアリポジトリ[7]から約 130 万件のクラス定義を収集し、そのうち 1 万件をランダムに抽出したクラス定義のメソッド名定義に用いられた語である。そのようなメソッドの定義に用いられた語（以降、メソッド語と呼ぶ）を定義されたクラス数の出現数に基づき、順位付けした表を表 2 に示す。

表 2 には、当該のメソッド語が定義として出現したクラス定義の上位 30 位までを示した。この表から分かるようにメソッド語の大半は、自然言語として一定の意味を持ちうる語彙により記述されることがわかる。すなわち、プログラム作成者は何らかの説明的意図を持ちつつ、メソッド名を命名している。しかし、*run* や *close* といった一般的な語でありながら、「走る」「閉じる」という一般的な意味では使われていない語が存在する。

また、*toString* や *hashCode* といったように、複数の語が合成されているものも多い。これらは単に複数の語が合成されているだけではなく、プログラム技法の専門的な概念に関わる言語表現に基づくメソッド語が、出現上位であることが分かる。

以上のように、経験を積んだプログラム作成者は、でたらめではなく、自然言語の意味に留意してメソッド名を命名していることが確認できた。しかし、同時に、そのような用法は、自然言語の一般的な語用ではないように思われる。そこで、メソッド名の命名に使われるような語の語用が、プログラム経験者が親和性を持つ語彙に関して、アルゴリズム説明文章を題材に、2 節で行ったプログラミング言語の教科書によるプログラム概念と対照可能な表現方法で分析を試みる。

4. アルゴリズム説明文章の調査

必ずしもプログラム初学者ではない読者に対して、アルゴリズムを説明する場合の語用について、2 節と同様に共起ネットワークの作成を試みる。まず、対象としたのは、アルゴリズム事典[8]の見出しと、完全一致する見出しページを持つ日本語版 Wikipedia[9]の 91 種類のアルゴリズムの説明ページである。この調査対象の文章に対して、2 節と同じ品詞制限の条件で共起ネットワークの作成をしたものが図 2 である。ただし、アルゴリズムの説明の文章量が大きく、共起ネットワークの計算対象とする語は 30 回以上の出現としている。Jaccard 係数は図 1 と同様に 0.1 以上であるものを関係性ありとしている。

図 1 と図 2 を対照比較してみると、それぞれの共起ネットワークの部分グラフが示す概念の説明対象が異なってい

ることが分かる。例えば、図 1 および図 2 の両方に出現する語「変数」では、前者がプログラミング言語の言語仕様における「変数」の概念導入であることに対して、後者では、「変数」の結びつきが、「確率」や「正規」といった、アルゴリズムの説明に確率や統計の理論が導入されることが部分グラフの差として示される。実際、調査対象とした 91 種類のアルゴリズム説明ページには、「モンテカルロ法」や「t 分布」といった説明が含まれている。このように、共起ネットワークによる文章で説明される概念の表現は、少なくとも記述内容を反映することが確認できる。

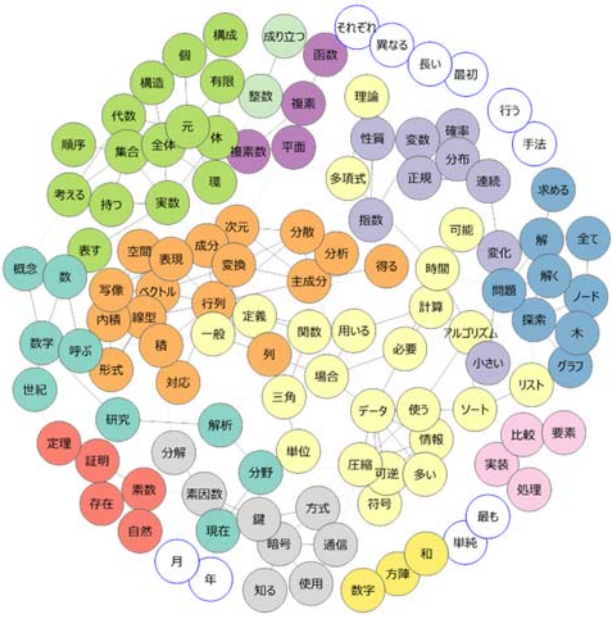

図 2　Wikipedia の 91 種類のアルゴリズム説明から得られた共起ネットワーク
Figure 2 Word co-occurrence network constructed from explanatory texts for ninety one algorithms

次に、表 2 に示されたようなメソッド語に選ばれるような語彙は、プログラム作成において参考にされる説明内に出現しうるかを検討するため、共起ネットワークを作成する文章の長さを調整することにより、共起ネットワークのノードとして出現させる語の出現数を調整し、共起ネットワークを比較・対照することを考える。

具体的には、日本語版 Wikipedia のソートアルゴリズムの説明ページのうち、「シェルソート」と「マージソート」についての説明ページの文章部分について、共起ネットワークを作成した。図 3 に「シェルソート」と図 4 に「マージソート」のそれぞれの説明ページの共起ネットワークを示す。　図 3 および図 4 の作成において、ノードとして選択される語の品詞情報による限定基準は図 1 および図 2 の場合と同じであるが、相対的な出現数による限定は異なる。具体的には、単一アルゴリズムの説明文数が少なくなるため、両図とも出現数の 2 以上の出現語に限定した。

さらに、図 1 で示したプログラミング言語の導入に関わる概念と、図 3 及び図 4 で示した個別のアルゴリズムの概念との対照比較をより詳しく見る基準となる長さの説明文章の語彙ネットワークも作成した。具体的には、図 2 で示した 91 種類のアルゴリズム説明文では、文章の長さが長大になってしまうため、シェルソート、マージソートに、さらにクイックソートやヒープソート等の 5 種類のソートアルゴリズムの説明ページを加え、計 7 種類のソートアルゴリズムについて、ソート系アルゴリズムの説明文章を作り、そこから共起ネットワークを作成した。ソート系アルゴリズムの説明から作成した共起ネットワークを図 5 に示す。

図 3 のシェルソートでは、図 1 の共起ネットワークでは出現数制限で現れず、図 5 に現れる「整列」、「ソート」、「挿入」、「間隔」、「グループ」といった語が共起ネットワーク上に出現している。

他方、図 4 のマージソートでも、図 1 では共起ネットワークの出現数制限で現れず、図 5 に現れる「マージ」「再帰」「整列」「ソート」といった語が共起ネットワーク上に出現する。当然のようにも思われるが、「ソート」「整列」は、図 3 および図 4 の両者に出現する。

参考までに、2 節の調査対象の教科書における「ソート」の出現数は 10 回、「挿入」の出現数は 5 回であり、対象の文章に語が出現しないのではなく、概念表現として表現した図 1 の共起ネットワークの作成制限により出現していないにすぎない。しかし、一定量の文章の中で説明しうる概念には上限があるはずであり、一定の共通性をもつ基準下で作成された共起ネットワークの対照比較には意義があるものと考える。

ソート系アルゴリズムに関わる概念を形成する部分グラフ要素になる語が、表 2 のメソッド語とどう関わるかについても調査した。表 2 は 1 万例のクラス定義数の上位 30 件のみしか掲載していないが、上位 1000 位までに、上の各ノードの語の英訳に相当するメソッドが対応しないかを人手で調査し、最も上位の語を対応付けると以下のようになる。

「挿入」: *insert*　（99 位）
「整列」、「ソート」: *sort*　（354 位）
「入れ替え」: *reverse*　（487 位）
「間隔」: *distance*　（504 位）

上記では、例えば「間隔」の訳語である、*interval* や *gap*, *space* といった語は表 2 の調査で得た上位 1000 語の中には存在せず、上位 1000 語のメソッド語の中では 504 位に *distance* という語が存在したことを記している。

ここで、図 4 および図 5 の両図に出現する「再帰」という語に相当する *recourse* は上位 1000 語のメソッド語の中に存在しなかった。また、「グループ」という語に相当すると考えられる *group* を包含するメソッド語は、getGroup（190 位），getGroupId （ 556 位），getGroups （ 907 位），isProcessGroupAlive（953 位）， killProcessGroup（978 位）と上位 1000 語の中に 5 つの定義が存在する。

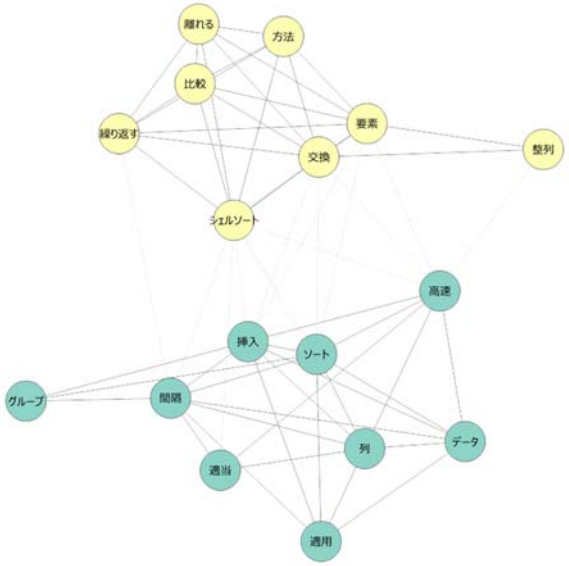

図 3 シェルソートの説明から得られた共起ネットワーク
Figure 3 Word co-occurrence network constructed from an explanatory text for the shell sort algorithm

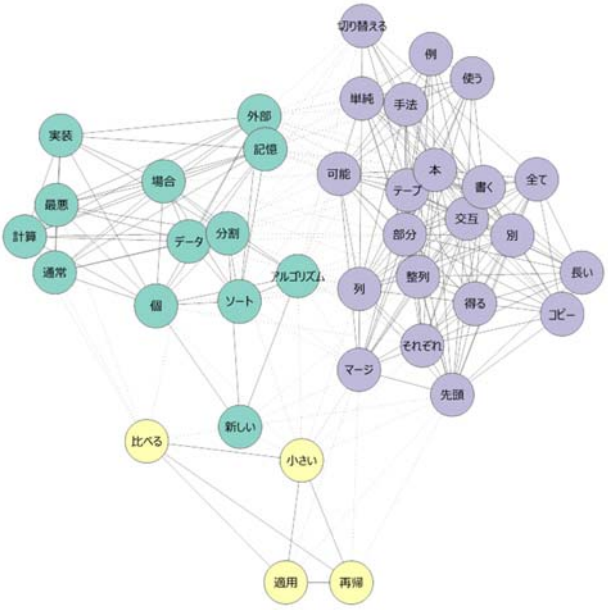

図 4 マージソートの説明から得られた共起ネットワーク
Figure 4 Word co-occurrence network for an explanatory text for the merge sort algorithm

このような、プログラム説明文書内の出現語彙と表 2 に部分例を示したメソッド定義との明確な関係は、現在のところ定義付けできていない。この定義付けについては、自動対応付けを主眼に検討を続けている。

以上のような本稿での調査は、プログラム教育の経験をもつ著者らが、試行錯誤をしつつ、教科書を中心とするプログラム説明文書を人手整理する上で知識整理に、テキストマイニング手法を導入することを試行した実践報告の位置づけとなる。具体的には、図 1 で示された語の関係性に親和性をもったと仮定しても、個別のアルゴリズムを説明した文章を、語の関係性のレベルで困難であろうことを、共起ネットワークによる概念表示の図 1、図 3、図 4 に対する対照・比較できることを提示した。

明確な基準を示すことはできなかったが、プログラミング言語の習得から個別アルゴリズムの習得につなげるためには、語の用法的には、表 2 のメソッド語の位置づけ、および図 5 のように特定の処理機能を持つアルゴリズムをまとめて整理し、基準となる語と語間関係を検討することが手掛かりになるだろうと考えている。

また、共起ネットワークの部分グラフが、アルゴリズム、モデル化あるいはデータ構造のどの概念とどのように関わるかのさらに詳細な分析は今後の課題である。

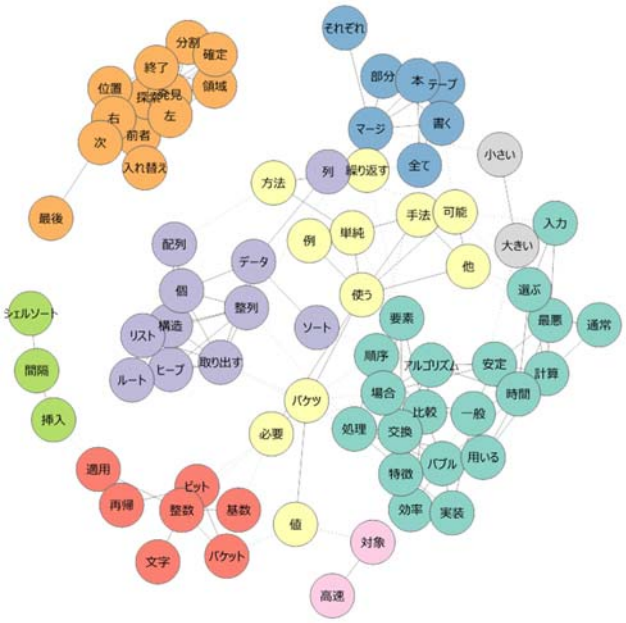

図 5 ソート系アルゴリズムの説明から得られた共起ネットワーク
Figure 5 Word co-occurrence network for explanatory texts for sort algorithms

5. おわりに

本稿では、テキストマイニングの 1 手法である共起ネットワークを概念表現として導入し、プログラミング言語の導入に使われる教科書によりプログラムの基礎概念がどのように表現されるかを、自然言語の共起ネットワークを使って概観した。次に、プログラム経験を積んだプログラム作成者が､それぞれ､どのような自然言語の表現を用いて、自らのプログラム理解を表明・説明するかをオープンソースの Java プログラムのクラス定義でメソッド名として使われる語を調査した。

以上の調査を前提に、アルゴリズムの説明文書を題材に、共起ネットワークを概念表現として見る観点から、説明文書の書き手がどのような語を使ってアルゴリズムの説明を行っているか、その語用について調査した。

結果として、個別アルゴリズムの説明に使われる語の語用は、プログラム学習者が親和性をもつ語彙とは異なる可能性があることを、共起ネットワークによる概念表現の対照比較により示した。

本稿の調査による知見から、多様なプログラム事例を学ぼうにも、その説明文章で使われる技術語彙が、プログラミング言語の学習段階のそれとは差異があることを示唆している。すなわち、専門的なプログラム作成を学ぶためには、プログラミング言語を学んだだけでは不十分で、実際にプログラムを作成し、そのプログラム部分の機能を示す適切な語彙を身に着けていなければ、プログラム作成の概念をその説明文章からのみでは理解できない可能性がある。

今後は、プログラム説明文書に対して、分析をさらに深め、より実効的なプログラム作成の学習・教育に資する説明法や、学習者のプログラム理解状態の評価方法に有益に活かしていきたい。

謝辞

本研究は JSPS 科研費(基盤(C)課題番号 15K01100)の助成を受けたものです。

参考文献

[1] 高橋麻奈:『やさしい C 第 2 版』,ソフトバンククリエイティブ, 2003
[2] 樋口耕一: テキスト型データの計量的分析 —2 つのアプローチの峻別と統合— 『理論と方法』, 数理社会学会, 19(1), pp.101-115, 2004
[3] 阪口祐介・樋口耕一:「震災後の高校生を脱原発へと向かわせるもの —自由回答データの計量テキスト分析から—」, 友枝敏雄編『リスク社会を生きる若者たち —高校生の意識調査から—』 , 大阪大学出版会, pp.186-203, 2015
[4] MeCab: http://taku910.github.io/mecab/ (参照 2018/01/25)
[5] M. Resnick et al.: Scratch: Program for All, Communication of ACM, 52-11, pp.60-68, 2009
[6] J.Gosling et al.: The Java language specification, third edition, Addison-Wesley, 2005.
[7] M. Allamanis et al.: Mining Source Code Repositories at Massive Scale using Language Modeling, The 10th Working Conference on Mining Software Repositories, IEEE, pp.207-216, 2013
[8] 奥村晴彦他: 『Java によるアルゴリズム事典』. 技術評論社, 2003
[9] Wikipedia 日本語版: https://ja.wikipedia.org/ (参照 2017/10/04)