Vol.2021-SE-209 No.9
2021/11/25

オンラインジャッジシステムにおける問題文の類似度調査

新濱 遼大[1,a)]　槇原 絵里奈[2,b)]　小野 景子[2,c)]　幾島 直哉[1,d)]　山川 蒼平[1,e)]

概要：オンラインジャッジシステム（OJS）には多種多様な問題が収録されており，プログラミング学習者の自学自習を支援することができる．一方で，学習者は多くの問題から自身のプログラミング能力や目的に合わせた問題を選択することが困難であり，学習に対するモチベーションの低下が危惧される．そこで OJS の問題文に着目し，問題文の類似度を算出することで学習者の問題選択支援を目指す．本研究では類似度を算出することで，問題選択における新たな指標となりうるか調査する．問題選択における指標として類似度を用いることが可能になれば，タグや難易度など問題選択における指標がない場合でも容易に問題を選択できる可能性がある．また，タグや難易度と組み合わせることで問題の選択肢がより縮小することが期待される．本研究では類似度を算出するために，まず自然言語処理である word2vec と doc2vec により問題文の分散表現を獲得し，評価を行った．結果として，問題文間の類似度は問題選択における新たな指標になる可能性を示した．

Similarity Investigation Between Problems in Online Judge System

1. はじめに

IT 産業の拡大によりプログラミン学習者が増加傾向にあり，初学者を対象とした講義や学習サイトが増加している．大学においては学生がプログラミング言語の特性や文法，機能について主体的に学ぶためにプログラミング演習が開講されている．プログラミング演習において学生は教員が用意した課題に対して個別に取り組み，疑問点や行き詰まりを教員や TA(Teaching Assistant) に質問することで解決する．そのため，プログラミング演習は疑問点や行き詰まりを迅速に解決できる点で，初学者の学習環境に適していると考える．しかし，プログラミングの基礎を学習した学習者に対して，プログラミング能力やアルゴリズムに対する理解を向上させる学習法は確立していない．そこで本研究では，プログラミング能力やアルゴリズムに対する理解を向上させる学習法としてオンラインジャッジシステム（Online Judge System，以下 OJS）に注目する．

OJS とはオンライン上でソースコードを送信すると自動でコンパイル・テスト・採点が行われ，結果を閲覧できるシステムである [1]．OJS は他の学習者が提出したソースコードを閲覧可能である．つまり，解法がわからない場合に他者のソースコードを参照することで解法の理解に繋がると考える．また，OJS に掲載される問題は難易度や解法に用いるアルゴリズムが多種多様である．以上の理由から，学習者は OJS を利用することでプログラミング言語やアルゴリズムの理解を深める自学自習が可能になると考える．しかしながら，OJS には多種多様な問題が存在するためプログラミング学習を目的に利用する学習者は問題選択が容易でない．学習者が適切な問題を選択できない場合，OJS の利用やプログラミング学習そのものに対するモチベーションが著しく低下する可能性がある．一部の OJS はユーザの問題選択を支援するためにタグや難易度を付与している．しかし，膨大な問題に対してタグや難易度のみを用いた絞り込みは困難である．例えば，Codeforces では難易度が 1000，タグが “implementation” の問題は 130 問以上ある．したがって，タグと難易度のみを基に学習者に問題提供を行うことは困難なため，新たな指標が必要と考える．そこで，OJS における学習者の問題選択を支援する

[1] 同志社大学理工学部研究科
Faculty of Science and Engineering, Doshisha University, Kyotanabe-shi, Kyoto 610–0394, Japan
[2] 同志社大学理工学部
Graduate School of Science and Engineering, Doshisha University, Kyotanabe-shi, Kyoto 610–0394, Japan
[a)] shinhama.ryota@mikilab.doshisha.ac.jp
[b)] emakihar@mail.doshisha.ac.jp
[c)] kono@mail.doshisha.ac.jp
[d)] ikushima.naoya@mikilab.doshisha.ac.jp
[e)] yamakawa.sohei@mikilab.doshisha.ac.jp

ため問題文に注目する．

本研究では問題の固有情報である問題文の類似度を算出することで，学習者の問題選択支援を目指す．算出した問題文間の類似度はタグや難易度に加えて，問題選択における新たな指標になると考える．本研究では word2vec と doc2vec を用いて，文章間の類似度を算出する．プログラミング学習において，数値や記号が問題文の類似性に与える影響は少ないと考える．そこで，文章に出現する単語のみに着目して類似度を算出するために単語の分散表現を獲得する手法である word2vec を用いる．しかし，類似度を算出する問題文で共通の単語が多く出現する場合，文章間の類似度が全て高くなる可能性がある．そのため文章の分散表現を獲得する手法である doc2vec を用いて類似度を算出する．2 つの手法を評価することで，プログラミング課題における文章間の類似度算出方法に対する知見を増やすことが可能になる．

本論文の構成は以下の通りである．まず 2 章において本研究の対象となる OJS の概要と OJS の教育現場への活用例を述べる．次に 3 章では本研究の基盤技術である word2vec と doc2vec について述べる．4 章では OJS における問題文間の類似度を算出する提案手法について述べ，5 章では 4 章で述べた手法を用いることで算出した類似度の結果および考察について述べる．最後に 6 章で本研究で得られた知見についてまとめを述べる．

2. オンラインジャッジシステム

2.1 概要

OJS とはユーザから提出されたプログラムをコンパイル・実行し, 複数のテストケースや検証器を用いて, 正確性や性能を自動評価し, ユーザへフィードバックするものである [1]. 代表例として，国内では AIZU ONLINE JUDGE[*1]や AtCoder[*2]，国外では TopCoder[*3]や Codeforces[*4]などが存在する．OJS では定期的に競技プログラミングやプログラミングコンテストが開催される．プログラミングコンテストでは複数の参加者が同時刻に同じ問題を解いていき，正解問題数や解答時間等に応じてユーザの順位付けやレーティングが行われる [2].

OJS 内の競技プログラミングやプログラミングコンテストで使用された問題は，後に OJS サイトに掲載されることが多く，ユーザはいつでも閲覧可能である．したがって，ユーザはコンテストやレートに関係なく，好きなタイミングで自由に問題に着手し，自身のプログラミングやアルゴリズム学習を進めることも可能である．

本研究ではコンテスト等ユーザの活動が活発で，各問題にタグと難易度が付与されている Codeforces を調査対象とする．

2.2 Codeforces

Codeforces とはロシア発祥のプログラミングコンテストサイトであり，2019 年時点での登録者は 6 万人以上，問題数は 6000 問以上の世界最大規模の OJS である．問題は番号とアルファベットによってまとめられている．例えば，問題番号 1 には A から C の 3 問が含まれており，A が最も簡単で B，C と進むにつれ複雑な問題となり，問題によっては J まで含まれる場合もある．また，多くの OJS において API が提供されており， Codeforces もユーザ情報や問題の情報など，多くのデータが API によって取得可能である[*5]．Codeforces は各問題に難易度とタグが付与されている．Codeforces では問題の難易度を 800 から 3500 までの 100 刻みの値で表し，問題には 37 種類のタグのどれか一つ以上が付与されている．タグの例として “dp(動的計画法)”，“string(文字列操作)”，“sorting(ソート)” などが存在する．また，タグは 1 問あたり複数付与されている場合もあり，例えば 3C の問題[*6]には “brute force(全探索アルゴリズム)”，“implementation”，“games” の 3 つのタグが付与されている．

2.3 プログラミング教育における OJS の活用

教員の採点コストの削減や結果に対する評価・フィードバックの自動化などを目的に，複数の教育機関において演習にオンラインジャッジシステムが開発，導入されている．岩本らは教員の採点・評価コストの削減，学生のコピーアンドペーストの防止を目的に，不正コピー検出手法を備えたオンラインジャッジシステムをプログラミング演習へ導入した [3]．岩本らが学生へアンケート調査を行った結果，多くの学生が OJS における Web 上でのプログラムの提出やフィードバック機能を利便性が高いと答え，さらに主体的な自学自習にも繋がったことが分かった．また，Kurnia らは OJS における採点は，公平性や効率の面から，教員が手作業で行うより優れていると述べている [4]．Zhou らは従来のコンパイル結果やテスト通過率に加え，コードの品質や他提出ソースコードとの類似度など，各学生に対しより詳細なフィードバックを与える OJS を開発した [5]．以上より，OJS をプログラミング学習へ導入することで，教員は採点コストを削減でき，学生は自身が提出したソースコードに対する評価やフィードバックをすぐに得られるため，学習に対するモチベーションの維持に繋がると考える．

また，既存の AOJ に掲載されている問題やソースコードそのものをプログラミング教育へ利用する取り組みも存在する．中川らは OJS における他ユーザのソースコード

[*1] https://onlinejudge.u-aizu.ac.jp/home
[*2] https://atcoder.jp/
[*3] https://www.topcoder.com/
[*4] https://codeforces.com/
[*5] https://codeforces.com/apiHelp
[*6] https://codeforces.com/problemset/problem/3/C

が閲覧できる点に着目した [6]．そして，既存ユーザの解答履歴を分析し，ユーザが提出した誤答ソースコードに対し，類似しているがより質が高く，正答へ繋がるソースコードを提示するシステム TAMBA を提案した．TAMBA を用いた実験の結果，学習者は闇雲にソースコードを参考にするより，TAMBA から提示されたソースコードを参考にしたほうが，誤答の原因発見に役立つと述べた．吉村らは，既存の OJS の問題及び各問題に対し Accept されたソースコードを解答例として収集し，演習で扱う課題に類似する問題及び解答例を提示するシステム WOJ Recommender を提案した [7]．吉村らは学生は課題に類似した問題を参考にすることで，課題を解くにあたっての難易度が下がり，WOJ Recommender を利用した学生の課題正答率が優位に上がったと報告した．

以上より OJS を適切に分析し，学習者の能力に応じた問題を提供することは，学習者のプログラミング教育の向上に有効であることが考える．

3. 基盤技術

3.1 word2vec

word2vec とは深層学習によって単語の分散表現を獲得する手法である [8]．word2vec における単語の分散表現では，周辺単語を用いて単語をベクトル空間上に表現する．そのため，word2vec では単語の意味を定量的に評価可能である．

word2vec には Continuous Skip-Gram Model(skip-gram) と Continuous Bag-of-Words Model(cbow) の 2 つのモデルが存在する．skip-gram の概念図を**図 1** に示す．図 1 より，skip-gram では入力層で 1 つの単語を与え，入力した単語の周辺を出力していることがわかる．つまり，skip-gram とは中心の単語から周辺の単語を予測するモデルである．次に cbow の概念図を**図 2** に示す．図 2 より，cbow ではある単語の周辺単語を入力として中心単語を出力していることがわかる．つまり cbow とは周辺の単語から中心の単語を予測するモデルである．cbow は skip-gram と比較し精度が低いことが報告されている [9]．

3.2 doc2vec

doc2vec とは，word2vec を拡張しすることで文章に対して分散表現を獲得可能な手法である [10]．従来は，文章に対して Bag-of-Words や TF-IDF を用いて分散表現を獲得していた [11], [12]．しかし，Bag-of-Words や TF-IDF は単語の出現頻度や共起から分散表現を獲得するため，文章内の語順を考慮していない．doc2vec は語順を考慮しベクトル空間に表現するため，Bag-of-Words と比較して単語の意味を評価可能な点が優れている．

doc2vec は 2 つの手法から文章の分散表現を獲得可能である．1 つ目が word2vec の skip-gram に対応する Distributed Bag-of-words(dbow) である．dbow はある単語から周辺単語を予測する手法である．dbow の特徴は, Bag-of-Words と同様に単語の語順を考慮せずに学習を行うため，効率的に計算可能な点である．2 つ目は word2vec の cbow に対応する Distributed Memory(dmpv) である．dmpv は文章 ID と複数の単語を入力として次に出現する単語を予測する過程を繰り返すことで，文章の分散表現を獲得する手法である．dmpv は単語の語順を考慮するため，dbow と比較して精度が高いことが報告されている [13]．

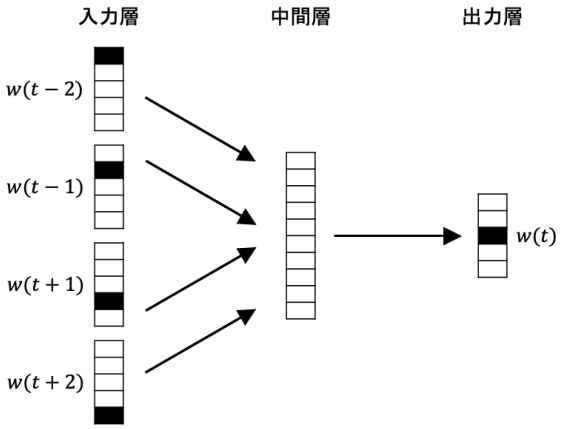

図 1 skip-gram の概念図
Fig. 1 Conceptual diagram of skip-gram.

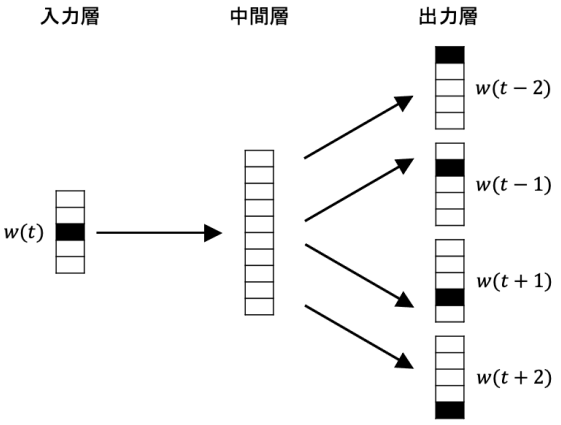

図 2 cbow の概念図
Fig. 2 Conceptual diagram of cbow.

4. 文章間の類似度算出手法

4.1 概要

本研究では OJS における問題文の類似度を算出することで, 学習者の問題選択支援を目指す．そのために, word2vec と doc2vec を用いて文章の分散表現を獲得し，文章間の類似度を算出する．なお，文章の類似度を算出するために本研究ではコサイン類似度を使用した．

word2vec と doc2vec のモデル作成には自然言語処理の分野で一般的に使用される python のライブラリである gen-

sim[*7]を使用した．word2vec のモデル作成には，wikipedia から収集した英単語に対してクリーニング処理を施したコーパスである text8[*8]を使用した．text8 は前処理済みのデータセットのため使用が容易であることから採択した．また，doc2vec のモデル作成には Codeforces から作成したデータセットを使用した．

4.2 データセット

本研究では，OJS のうち Codeforces を対象としてデータセットを構築した．データセット作成のために 2021 年 8 月時点で取得可能であった 1〜1553 のコンテストを対象に 6741 問のデータを取得した．取得したデータの種類は，問題 ID，問題文，タグである．

4.3 word2vec を用いた類似度算出手法

word2vec を用いた類似度算出過程を以下に示す．

Step1 text8 を使用し word2vec のモデルを作成
Step2 データセットの前処理
Step3 作成したモデルを用いて単語の分散表現を獲得
Step4 単語の分散表現を基に文章の分散表現を獲得
Step5 文章の分散表現からコサイン類似度を算出

Step1 において skip-gram を使用し，次元数は 300 次元としてモデルを作成した．また，ウィンドウサイズは 5 とし，訓練回数は 10 とした．なお，モデル作成において 5 回未満登場する単語は棄却した．その他のパラメータに関しては，デフォルト値を使用した．

次に Step2 では Codeforces から作成したデータセットを使用するにあたり，前処理として単語の正規化を行った．単語を正規化することで，数値や不等号や演算子などの記号を削除を可能にした．しかし，プログラミング課題には数値や記号が含まれることが多く，データセットの問題文にも多く含まれていた．したがって，今後は単語の分散表現のみではなく数値や記号の分散表現を獲得する手法を考案する必要があると考える．また前処理では，自然言語処理のツールキットである NLTK[*9]を用いてストップワードを除去した．

そして Step3，4 において作成したモデルに問題文を入力することで，問題文に含まれる全単語の分散表現を獲得する．しかし，word2vec は未知語の分散表現を獲得することができない．未知語とはモデルに登録されていない語句や表現である．そのため，今後は未知語の分散表現を獲得可能な手法である fasttext の使用を検討している [14]．単語の分散表現獲得後は，Simple Word-Embedding-based Methods(swem) を使用することで文章の分散表現を獲得する．swem とは単語の分散表現から文章の分散表現を獲得する手法であり，複数のデータセットにおいて既存の CNN モデルと同等以上の性能であることが明らかである [15]．なお，本研究では swem で提案されている手法のうち，単語ベクトルの平均から文章の分散表現を獲得する swem-aver を使用した．swem-aver では文章に含まれる単語の分散表現に対して average pooling を行う手法である．

最後に Step5 において問題文の組み合わせに対して，コサイン類似度を算出することで問題文間の類似度を算出する．

4.4 doc2vec を用いた類似度算出手法

doc2vec を用いた類似度算出過程を以下に示す．

Step1 データセットを使用し doc2vec のモデルを作成
Step2 作成したモデルを用いて文章の分散表現を獲得
Step3 文章の分散表現からコサイン類似度を算出

Step1 において doc2vec のモデル作成に dmpv を使用し，次元数は 64 次元とした．モデル作成においては，データセットが小さいため単語の棄却は行わなかった．また，エポック数は 20 とし，その他のパラメータに関してはデフォルト値を使用した．

Step2，3 において Step1 で作成したモデルを使用することで問題文の分散表現を獲得し，文章間の類似度を取得した．

5. 結果と考察

5.1 類似度算出結果

ランダムに抽出した 50 問のデータに対して，word2vec と doc2vec より算出した問題文間の類似度を**図 3** に示す．また，**表 1** に 5 つの問題文の組み合わせに対して word2vec と doc2vec により算出した類似度を比較した表を示す．図 3 より，手法として word2vec を用いた場合は第一四分位数が 0.7 を超えており類似度が高い傾向にあることがわかる．これは問題文に含まれる単語が共通することが多いことに起因すると考える．共通する単語を確認したところ，型指定や変数が多く問題文に含まれていることが確認できた．本研究では word2vec のモデル作成に wikipedia から取得したデータセットを使用した．よって，型指定や変数などプログラミング課題特有の単語に対して適切な分散表現を獲得できなかったと考える．今後はプログラミング課題特有の単語に対応できるデータセットを構築し，検証を行う必要がある．また，swem-aver では文章に含まれる全単語に対して average pooling を行うため，希少語と頻出度の影響が小さくなる．つまり，問題文に含まれる単語数が多くなるにつれ文章の分散表現に大きな差が生じず，算出し

*7 https://radimrehurek.com/gensim/
*8 http://mattmahoney.net/dc/text8.zip
*9 https://www.nltk.org/

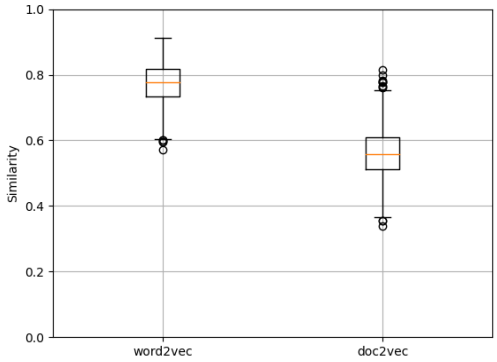

図 3 word2vec と doc2vec を用いた類似度算出結果
Fig. 3 Result of similarity calculation by word2vec and doc2vec.

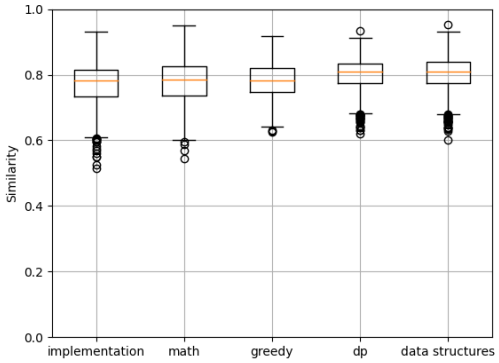

図 4 word2vec を用いたタグごとの類似度算出結果
Fig. 4 Calculation results of similarity for each tag by word2vec.

表 1 word2vec と doc2vec により算出した類似度の比較
Table 1 Comparison of similarity by word2vec and doc2vec.

	類似度算出手法	
問題 ID	word2vec	doc2vec
1042C と 1523D	0.904374	0.566608
1033D と 538B	0.866826	0.667299
1041C と 1154D	0.784892	0.587641
380C と 1283A	0.722142	0.544207
660B と 1505I	0.607825	0.550637

た類似度が高い傾向を示した可能性がある．

また，図 3 より doc2vec を用いた手法は word2vec を用いた手法と比較して，算出した類似度が低い傾向にある．これは doc2vec が数字や記号を考慮して類似度を算出可能な点が起因すると考える．全ての単語を考慮して類似度を算出する doc2vec は，文章の意味的類似度をより正確に算出した可能性がある．

表 1 より手法によって算出する類似度に差が生じることがわかる．そこで，本研究では被験者実験を実施することにより文章間の類似度を定性的に評価する．

5.2 タグごとの類似度算出結果

問題文に付与されているタグ情報は，問題文の類似度に影響を与えている可能性がある．そこで，タグごとにランダムで 50 問ずつ抽出したデータに対して，問題文間の類似度を算出する．なお，対象とするタグは付与された回数が多い “implemention”，“math”，“greedy”，“dp”，“data structures” である．word2vec を用いて問題文間の類似度を算出した結果を**図 4** に示す．同様のデータに対して doc2vec より算出した問題文間の類似度を**図 5** に示す．図 4 と図 5 から手法の差異に関係なく “implemention” と “math” はデータのばらつきが大きいことがわかる．また，“greedy” と “dp” はデータのばらつきが小さいことがわかる．“implemention” と “math” は問題のカテゴリを表すのに対して，“greedy” と “dp” はアルゴリズム情報を表している．つまり，アルゴリズム情報を基に付与されたタグ同士の問題文は類似度が高い可能性がある．

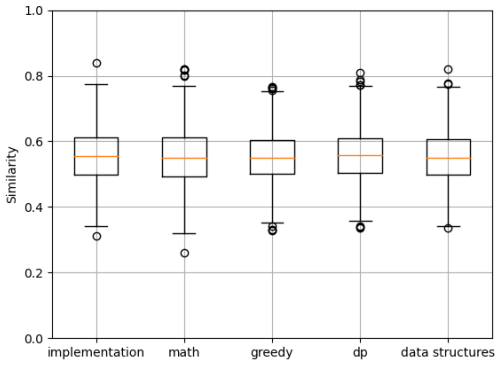

図 5 doc2vec を用いたタグごとの類似度算出結果
Fig. 5 Calculation results of similarity for each tag by doc2vec.

5.3 被験者実験による定性的評価

word2vec と doc2vec より算出した類似度を定性的に評価するため，被験者実験を実施した．それぞれの手法で類似度が高い問題の組み合わせ 5 組と低い組み合わせ問題の 5 組を課題として，類似しているかを 5 段階で評価する被験者実験を行った．評価基準は 2 つの問題が似たような知識で解けるかとした．被験者はプログラミング経験が 3 年以上ある 20 代学生 10 名である．実際の OJS 利用環境を想定して，被験者には入力条件や出力条件を配布した．また，同様の理由から被験者実験中にブラウザ検索を可能とした．被験者実験の結果を**図 6** と**図 7** に示す．図 6 の A1〜A5 は word2vec を用いて算出した類似度が高い組み合わせを示し，B1〜B5 は類似度が低い組み合わせである．同様に図 7 の C1〜C5，D1〜D5 はそれぞれ doc2vec を用

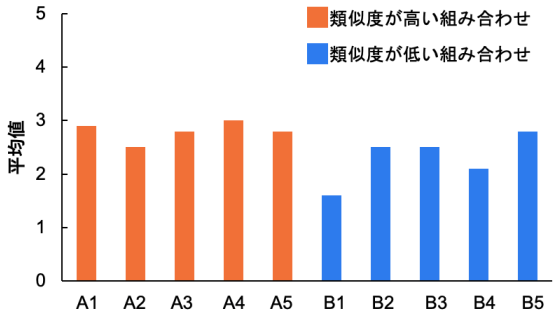

図 6 文章の類似度を定量的に評価した実験結果（word2vec）
Fig. 6 Quantitative evaluation results of document similarity(word2vec).

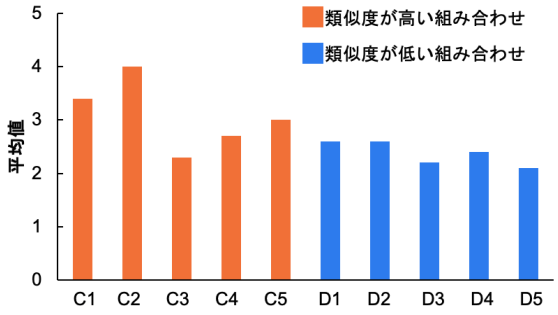

図 7 文章の類似度を定量的に評価した実験結果（doc2vec）
Fig. 7 Quantitative evaluation results of document similarity(doc2vec).

いて算出した類似度が高い組み合わせと低い組み合わせである．図 6 と図 7 より算出した類似度が高い組み合わせは，低い組み合わせと比較して類似度が高いことがわかる．特に類似度が高い組み合わせである C1 と C2 に関しては付与されているタグから，同じアルゴリズムを使用していることがわかった．つまり，doc2vec より算出した類似度とタグを組み合わせることにより学習者に最適な問題を提案できる可能性がある．また，図 6 と図 7 より類似度が 3 を超えない組み合わせが多く存在することがわかる．類似度が 3 を超えない組み合わせを確認すると被験者ごとに解答した類似度に大きな差が生じていた．これは類似度を評価する基準が曖昧であることが原因であると考える．また，被験者に確認したところ実際にコードを書かないため評価が難しかったという意見が得られた．定性的評価における doc2vec を用いて算出した類似度の評価が低い理由として，モデル作成時に学習させるデータセットが少ないことが起因すると考える．つまり，データセットが十分にないため適切な分散表現を獲得できなかった可能性がある．したがって，学習に使用するデータセットを増やすことで，精度が向上する可能性がある．

6. おわりに

本論文ではプログラミング能力やアルゴリズムの理解を向上させる学習法として OJS に注目した．OJS における学習者の問題選択を支援するために word2vec と doc2vec を用いて，問題文間の類似度を算出した．そして算出した類似度をグラフ化することで分析し，被験者実験により定性的に評価した．結果として，プログラミング課題特有の変数名や数値，記号を考慮して分散表現を獲得可能な doc2vec が高い評価を示した．したがって，本研究では学習者の問題選択における新たな指標になり得る可能性を示した．また，タグごとに類似度を算出した結果より共通のアルゴリズムが付与された問題文間の類似度は，高い可能性があることを示した．つまり，タグと問題文間の類似度を提供することで，学習者の問題選択が容易になる可能性がある．

今後はプログラミング課題特有の単語に対して，適切な分散表現を獲得できるデータセットの構築する．また，算出した類似度を難易度やソースコードなどのタグ以外の情報から評価することを検討している．そして，問題文間の類似度を問題選択における新たな指標とすることを目指す．

参考文献

[1] 渡部有隆ほか：オンラインジャッジの開発と運用-Aizu Online Judge，情報処理，Vol. 56, No. 10, pp. 998–1005 (2015).
[2] Ebtekar, A. and Liu, P.: An Elo-like System for Massive Multiplayer Competitions, *arXiv preprint arXiv:2101.00400* (2021).
[3] 岩本舞，中村真人，小島俊輔，中嶋卓雄ほか：不正コピー検出手法を備えたオンラインジャッジシステムの開発，情報処理学会論文誌教育とコンピュータ (TCE)， Vol. 1, No. 4, pp. 38–47 (2015).
[4] Kurnia, A., Lim, A. and Cheang, B.: Online judge, *Computers & Education*, Vol. 36, No. 4, pp. 299–315 (2001).
[5] Zhou, W., Pan, Y., Zhou, Y. and Sun, G.: The framework of a new online judge system for programming education, *Proceedings of ACM Turing Celebration Conference-China*, pp. 9–14 (2018).
[6] 中川尊雄， 藤原新， 畑秀明，松本健一ほか：プログラミング学習者向けソースコード提示システム TAMBA，ソフトウェアエンジニアリングシンポジウム 2016 論文集， Vol. 2016, pp. 34–41 (2016).
[7] 吉村涼矢，坂本一憲，鷲崎弘宜，深澤良彰ほか：プログラミング教育のための類題出題システムの提案，研究報告コンピュータと教育 (CE)， Vol. 2020, No. 3, pp. 1–6 (2020).
[8] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. and Dean, J.: Distributed representations of words and phrases and their compositionality, *Advances in neural information processing systems*, pp. 3111–3119 (2013).
[9] Jang, B., Kim, I. and Kim, J. W.: Word2vec convolutional neural networks for classification of news articles and tweets, *PloS one*, Vol. 14, No. 8, p. e0220976 (2019).
[10] Le, Q. and Mikolov, T.: Distributed representations of sentences and documents, *International conference on*

machine learning, PMLR, pp. 1188–1196 (2014).
[11] 岡田将吾，松儀良広，中野有紀子，　林佑樹，　黄宏軒，　高瀬裕，新田克己：マルチモーダル情報に基づくグループ会話におけるコミュニケーション能力の推定，人工知能学会論文誌，Vol. 31, No. 6, pp. AI30–E_1 (2016).
[12] 松尾豊，　石塚満：語の共起の統計情報に基づく文書からのキーワード抽出アルゴリズム，人工知能学会論文誌，Vol. 17, No. 3, pp. 217–223 (2002).
[13] Dai, A. M., Olah, C. and Le, Q. V.: Document embedding with paragraph vectors, *arXiv preprint arXiv:1507.07998* (2015).
[14] Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T.: Enriching word vectors with subword information, *Transactions of the Association for Computational Linguistics*, Vol. 5, pp. 135–146 (2017).
[15] Shen, D., Wang, G., Wang, W., Min, M. R., Su, Q., Zhang, Y., Li, C., Henao, R. and Carin, L.: Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms, *arXiv preprint arXiv:1805.09843* (2018).