

自習中の学生のための科目を超えた関連ページの発見

印部 太智^{1,a)} 丸山 一貴^{2,b)}

概要: 大学のカリキュラムは、1, 2 学年に基礎科目を履修し、3, 4 学年で応用科目を履修する。情報科学系の学科に入学する学生が興味をもつ AI やセキュリティなどについての内容は応用科目に該当する。そのため、基礎科目を学習中は、その科目と興味のある内容との関連性を想像できず、学習のモチベーションを維持できないと考えた。そこで、我々は授業資料に注目する。授業資料は、スライドやレジュメなどが考えられるが、特にスライドはそれぞれのページで内容がまとめられており、一目見ただけである程度内容を知ることができる。本研究では、スライド形式の資料を用いて自習中の学生が注目した箇所に基づき、学習済みあるいは今後学習する資料の中から関連する具体的なページを発見することを目的とする。関連するページを受け取った基礎科目を学習中の学生は、現在の科目の内容が興味のある科目に応用されていることを知り学習に対するモチベーションを高めたり、理解が難しい内容は学習済みの内容から改めて整理しなおすことができる。関連するページを発見するために、本研究では TF-IDF を用いてページ単位でスライドを読み込み、それぞれのページの重要な単語を求めた。TF-IDF に対して、時系列を重視し、正規化を行い、単語に対して重み付けを行う補正を適用する。スタンフォード大学コンピュータサイエンスコースの科目の Web ページから入手可能なスライド資料を用いて、学生にとって有効なページを発見できるか実験を行った。本論文ではその実験結果の中から興味深い結果を示す。

キーワード: TF-IDF, 授業資料, 単語分析, スライド, 学習環境, スコアリング

Finding Relevant Pages from All the Subjects for Students in Self-Studying

IMBE TAICHI^{1,a)} MARUYAMA KAZUTAKA^{2,b)}

Abstract: University curricula have the foundation subjects at the basic undergraduate students and the application subjects at the advanced undergraduate students. For example, AI and cybersecurity in which most basic undergraduate students are interested, are covered in the application subjects. For that reason, it is difficult for the basic undergraduate students to imagine the connection between the topics and the subjects and to keep the motivation. Therefore, we focused on the learning materials, particularly the slides. By using the slides, the students know the page's learning theme easily. Our proposal is to find the relevant pages from all the subjects, based on the page on which a basic undergraduate student just focuses. Providing the relevant pages based on the page of the learning materials helps the students in the basic undergraduate to enhance the one's motivation and the students in the advanced undergraduate to deepen the one's understanding. In this study, we use TF-IDF to discover the relevant pages and calculate the characteristic words. In addition, we introduce chronological order, normalization, and the term weighting to TF-IDF. This paper conducts an experiment using the materials available at Stanford University CS courses' website and shows the characteristic results and the feasibility of our proposal.

Keywords: TF-IDF, Learning materials, Word analysis, Slide, Learning environment, Scoring

¹ 明星大学大学院 情報学研究所
Graduate School of Information Science, Meisei University
² 明星大学 情報学部

School of Information Science, Meisei University
^{a)} 20mj003@stu.meisei-u.ac.jp
^{b)} kazutaka.maruyama@meisei-u.ac.jp

1. 背景

大学のカリキュラムは1, 2学年のときに基礎科目を履修し, 3, 4学年では応用科目を履修する。情報学科に入学する学生が興味をもつ AI やセキュリティなどに直接関わる科目の多くは, 応用科目になっていることが多い。基礎科目を学習中は, その学習が応用科目に発展することをイメージできず, 学生のモチベーションが維持できないと考えた。学生のモチベーションを維持するためには, 基礎科目を学習中に応用科目の内容を示すことが有効であると考えた。

本研究では図1のようなシステムを検討し, 学生のモチベーションの維持を図る。図1は学生がLMSに公開されている授業用のスライドを用いて自習中であると仮定し, その内容に基づいて学習済みや発展的な内容のページを示すシステムである。授業用のスライドは, ページごとにトピックを定めた説明があるため, 学生が一目で内容を把握できる。また, LMSの内部では特定の学年の資料だけではなく, 大学全体で公開された資料が蓄積され, 他学年の資料も利用できる。図1は, 学生が **tree** という単語に注目した場合である。このとき, 学習済みの科目からは二分木やディレクトリの構造に関するページを, これから学習するページからはコンパイラや自然言語処理の構文木に関するページを示している。これにより, 学生は **tree** という単語から現在の学習がカリキュラム全体でどのような位置付けなのか整理できる。

図1は, 以下の3ステップで構成できる。

ステップ1 学習中のページから単語(名詞)をすべて取得する。タイトルや箇条書きの文字として説明されていると考えるためである。名詞を用いることで, そのページの特徴的な単語が取得できる。

ステップ2 それぞれの単語に基づいて, その単語が含まれる全ての科目の全てのページに対してスコアリングを行う。

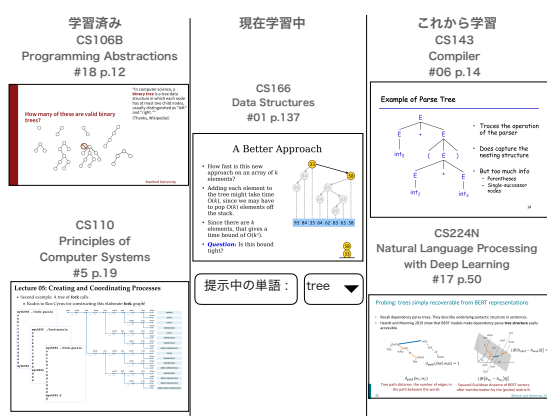


図1 システムが提示する内容の例 (CS106B[1], CS110[2], CS149[3], CS143[4], CS224N[5] から引用)

ステップ3 ステップ2のスコアに基づいて, 学生に対してページの提示する。

本研究の目的は, 図1に示したシステムを実現するための基礎研究として, ステップ2における, 単語を用いたすべての科目のページに対するスコアリングを行うものである。本研究では, スコアリングの手法としてTF-IDFを採用する。TF-IDFに対して授業資料の特徴を考慮するために, 時系列や重み付け, 正規化を導入し, スコアの補正を行う。これらのスコアに基づいて, 関連するページを学生に示す。これから学習するページが示されることで, 現在学習中の内容がどのように応用されるのか知り, 学習意欲の向上や興味のある分野の発見につながる。

2. 関連研究

2.1 科目の関連付け

Goncharowら[6]は, ACMとIEEEが公開しているカリキュラムガイドラインに対して授業資料のマッピングを行う手法を提案している。山本ら[7]は, シラバスとカリキュラムマップに基づいてこれらを結びつける手法を提案している。これらの手法では, 体系的に整理されたものを関連づける。これらの手法を活用することで, カリキュラム全体を俯瞰し, ある科目がどういった科目に発展するか発見できる。シラバスやカリキュラムマップはマクロな視点であり, 授業資料はミクロな視点であるため, 本研究の手法は, より具体的な内容を学生は関連づけることができる。

2.2 スライドの構造を重視する手法

Sajjacholapuntら[8]は, 演習問題の文章とスライドのページタイトル部分を重視する重み付けを行うことで, 演習問題と関連するページを発見する手法を提案している。この手法では, タイトル部分の単語を重視するために, 単語がタイトル部分に現れた場合にその出現頻度を重視する形でTF-IDFに重み付けしている。桐原ら[9]は学習中の学生が注目している授業スライドの1ページに対して関連するページを示す手法を検討している。この研究では, スライドが持つインデントの深さに応じて, そのページ内の単語の重要度(論文[9]中では寄与率と表現)を求めている。インデントの深さはページタイトルが1.0になるように重みを設定し, 0.5, 0.33, 0.25としている。これらの手法では, ページタイトルや箇条書きといったスライドの特徴に注目し, それらを重要視している。本研究とは, スライドの構造に注目し, スコアに補正を行う点が類似する。ページタイトルや箇条書きという構造的な特徴に加え, 授業内の説明の順序を重視する。

2.3 TF-IDFの改善

Satoら[10]はTF-IDFに補正を行い, 順序関係を持た

せる手法として Chronological incremental TF-IDF(以下、C-TF-IDF という。)を提案している。この手法を用いて、災害後の Web ニュースの単語の時間的な変化を分析している。また、Paik[11] は文書に含まれる単語数に基づいて、正規化する手法を提案している。本研究では、これらの手法を採用し、授業内の説明の順序を重視した補正やページごとの単語数の違いの正規化を行う。

我々の先行研究では、特定の科目における授業回ごとのスコアの変化について、TF-IDF と C-TF-IDF のスコアを比較した [12]。スライド形式の授業資料を対象とし、第一回から順番に TF-IDF と C-TF-IDF を計算した。その結果、C-TF-IDF が授業資料内での単語の出現時期を考慮した授業回を発見することができた。本研究では、この結果を踏まえて、C-TF-IDF をページ単位で適用する。

3. 提案手法

ここでは、第 1 章で述べたステップ 2 を実現するための手順について述べる。

3.1 スコアリング手法

本研究では、スライドを用いて自習中の学生に対し、科目を超えて過去や未来の関連するページを示すために、ページへのスコアリング手法を提案する。関連する科目を発見する手法として、第 2.1 節で述べた手法は、内容が抽象化されるため、学生が科目の詳細を把握しにくい。キーワードのようなものも想定できるが、キーワードの設定への負担やキーワードによって関連する科目の範囲が制限されると考えたため、採用を見送った。ページ単位で提示を行うことで、学生は一眼で科目の関係を俯瞰できる。

TF-IDF を用いてページ単位でスコアリングを行う。この手法を用いることで、ある単語が特に重要視されているページを発見することができる。TF-IDF 以外にも単語を用いる手法として、オントロジーや Word2Vec などが挙げられる。オントロジーは更新にかかるコストが大きいため採用を見送った。Word2Vec はスライドが持つ特徴の考慮が難しいと考えている。

一方、授業資料やスライドには、TF-IDF では検討しきれない、以下の特徴があると考えられる。

- 第 1 回から最終回まで連続になっていて、過去の内容に基づいて説明が展開されるなど、時系列が存在する。
- スライド自体の特徴としてページタイトルや箇条書き、文字への装飾があり、スライドの作成者が意図して強調することができる。
- 図形が主体のスライドと文字が主体のスライドがあり、ページ内の単語数が異なる。

本研究では、TF-IDF に対して、上述の特徴を重視する補正を取り入れスコアリングする。導入する補正として、以降、第 3.2 節で TF-IDF に時系列を導入、第 3.3 節では

授業スライドの特徴を考慮した重み付け、そして、第 3.4 節で単語数を考慮した正規化についてそれぞれ述べる。本研究では、TF-IDF の計算を行う段階で、分析のためにタイトルだけのページは除去する。タイトルだけのページはその後続くページのキーワードを発見する上で有効であるが、学生に提示する情報としては不足していると考えられる。

3.2 Chronological incremental TF-IDF の導入

本研究では、TF-IDF に対して時系列を考慮した補正を行う。TF-IDF に時系列を導入した手法として、本研究では C-TF-IDF[10] を採用する。この手法は、ある期間から特定の時期までの一定区間内の単語の変化を見ることに適しているため、本研究においても有効であると考えた。C-TF-IDF は式 (1) で定義される。ここで、 t_i は $TF-IDF(t_i, d_j)$ で計算するある 1 単語で、添字 i は文書 d_j 内の単語数を n とし、 $1 \leq i \leq n$ の範囲をとる。 d_j は、 t_i を含む特定の文書で、添字 j は文書数を N とし、 $1 \leq j \leq N$ の範囲を取る。 N は、総文書数である。 $TF(t_i, d_j)$ は、文書 d_j 内の単語 t_i の出現頻度を求める。 $DF(t_i, d_j)$ は、文書 d_j 出現時の単語 t_i の出現文書数を求める。

$$\text{Chronological Incremental TF-IDF}(t_i, d_j) = TF(t_i, d_j) \cdot IDF(t_i, d_j) \quad (1)$$

$$IDF(t_i, d_j) = \log\left(\frac{N_j}{DF(t_i, d_j)}\right) \quad (2)$$

文書数が増え、単語 t_i が出現する文書が増えると IDF の値は次第に減少する。この手法を用いることで、ある単語が初出のページや別の内容を扱った後に出現するページを重視できる。

3.3 スライドの特徴を重視した重み付け

授業用のスライドは以下のような要素を持つ特徴がある。

- スライドの上部にあるページタイトル部分。
- 箇条書き。
- 太字や色を付けるなど文字への装飾。
- ページ内の図や表。

本研究ではこのうち、ページタイトル、箇条書き、文字への装飾について重視する。図や表は、本研究では対象としていないが、図に代替テキストがある場合や表内のテキストが取得できる場合について検討していく必要がある。情報系の科目で用いられるソースコードについては、テキストが取得できるものを対象とする。ソースコードも単語単位でスコアを算出することで、実装以外を述べたページとの関連付けが可能になると考えている。実験に使用する科目の多くは、太字で重要な単語を装飾していたため、本研究では太字を対象とする。それ以外の装飾は今後検討する。

重み付けには式 (3) から式 (5) を用いる。ページタイトル

と太字に関する重み付けはそれぞれ式 (3) と式 (4) である。 $T(d_j)$ と $B(d_j)$ は、ページ d_j でタイトルや太字に含まれる単語の集合を求める。 $Title-DF(t_i, d_j)$ と $Bold-DF(t_i, d_j)$ は、単語 t_i が d_j までにタイトルや太字として出現した頻度を求める。タイトルや太字として重視すべきものは、C-TF-IDF で重視するものと同様であると考えた。そこで、重みを計算するページまでの出現頻度を求める。箇条書きに関する重み付けは式 (5) である。 $indent$ は、箇条書きのインデントにあたる添字で、深さは 4 までを考慮する。 w_{indent} はインデントの深さ $indent$ が 0 から順に 3.5, 2, 1.8, 1.5 とする。 $freq(indent, t_i, d_j)$ は、インデント $indent$ でページ d_j までの単語 t_i の出現回数を求める。箇条書きは、最も浅いインデントで重要な単語が出現し、深くなるごとに浅い部分を修飾する関係にあると考えた。インデントの深さに数値をつけ、ページ d_j までのある単語の深さごとの出現回数を求めることで、インデントが浅い箇所に多く出現した単語を重視する。

$$T_w(t_i, d_j) = \begin{cases} \frac{N_{d_j}}{Title-DF(t_i, d_j)} & t_i \in T(d_j) \\ 1.0 & t_i \notin d_j \end{cases} \quad (3)$$

$$B_w(t_i, d_j) = \begin{cases} \frac{N_{d_j}}{Bold-DF(t_i, d_j)} & t_i \in B(d_j) \\ 1.0 & t_i \notin d_j \end{cases} \quad (4)$$

$$I_w(t_i, d_j) = \log(\sum_{indent=0}^4 freq(indent, t_i, d_j) * w_{indent}) \quad (5)$$

式 (3) から式 (5) を用いて、C-TF-IDF にさらに重み付けする。式 (6) はそれぞれの重みを足し合わせ、正規化した C-TF-IDF と合わせる。重み同士を積算する方法も考えたが、重みが必要以上に影響を与えたため、本研究では足し算とした。この重みによって、資料を作成した教員がそのページで重視した単語を考慮することができる。

$$Weighted-C-TF-IDF = Sim_{norm}(Q, D) * (T_w + B_w + I_w) \quad (6)$$

3.4 TF-IDF の正規化

授業用のスライドのページには、以下のような種類がある。

- 図が多いページ。パラパラ漫画のように類似した図が連続するページも該当する。
- 図と文章を組み合わせたページ。
- 文章のみのページ。

これらに TF-IDF を適用した場合、図が多い場合は 1 つ 1 つの単語を重視し、文章を主とすると出現頻度が少ない単語を重視する。また、ソースコードも単語として扱い、TF-IDF の計算を行う。このとき、変数名として用いている単語は、ページ内の出現回数が多く、そのページにお

る TF-IDF のスコアが下がってしまう。これでは、本来重視したい単語を重視しきれないと考えたため、C-TF-IDF で求めたスコアに対して、ページ内の単語数に基づく正規化を行う。

正規化には Paik[11] の手法を用いる。この手法は TF-IDF をベースとし、対象となる資料の単語数に応じて正規化を行う。正規化に用いる計算方法は式 (7) から式 (15) で、式 (7) から式 (12) までは TF の正規化、式 (13) と式 (14) は IDF の正規化である。最終的なスコアは式 (15) によって算出される。

$$RITF(t, D) = \frac{\log_2(1 + TF(t, D))}{\log_2(1 + Avg.TF(D))} \quad (7)$$

$$LRTF(t, D) = TF(t, D) \times \log_2(1 + \frac{ADL(C)}{len(D)}) \quad (8)$$

$$BRITF(t, D) = \frac{RITF(t, D)}{1 + RITF(t, D)} \quad (9)$$

$$BLRTF(t, D) = \frac{LRTF(t, D)}{1 + LRTF(t, D)} \quad (10)$$

$$TFF(t, D) = w \times BRITF(t, D) + (1 - w) \times BLRTF(t, D) \quad (11)$$

$$w = QLF(Q) = \frac{2}{1 + \log_2(1 + |Q|)} \quad (12)$$

式 (7) は特に単語数が多い場合に、式 (8) は単語数が少ない場合に有効である。ここで登場する $ADL(C)$ は全ての文書の単語数の平均である。式 (9) と式 (10) は、式 (7) と式 (8) を用いて、 $0 < f'(x) < 1$ の範囲としたものである。式 (11) は、重み w によって式 (9) と式 (10) のバランスを取る。 w は式 (12) によって決定し、この式 (12) は検索時に使用する単語の集合や類似度を比較する際の単語の集合などの $|Q|$ によって決まる。本研究では、Paik が比較した中でもっとも有効とされている手法を採用し、 $|Q|$ のサイズについては 1 つの単語での計算を想定するため 1 とする。

式 (13) は平均出現頻度である式 (14) を用いて IDF の値を正規化する。式 (14) で登場する $CTF(t, C)$ は、すべての資料の中での単語 t の総出現数である。本研究では C-TF-IDF を採用するため、あるページまでに出現した単語 t の総出現数となる。式 (15) では、入力された単語の集合 $|Q|$ との類似度の計算を行うことで、正規化した値を算出する。

$$TDF(t, C) = IDF(t, C) \times \frac{AEF(t, C)}{1 + AEF(t, C)} \quad (13)$$

$$AEF(t, C) = \frac{CTF(t, C)}{DF(t, C)} \quad (14)$$

$$Sim_{norm}(Q, D) = \frac{\sum_{i=1}^{|Q|} TFF(q_i, D) \times TDF(q_i, C)}{\sum_{i=1}^{|Q|} TDF(q_i, C)} \quad (15)$$

4. 実験

提案手法によるスコアリングの有効性を検証するために、科目ごとにスコアを算出するシステムを作成し、評価実験を行った。実験には、Stanford University の CS コースの科目の Web サイトに公開されているスライドを用いた。本実験では、CS106B[1], CS106L[13], CS107[14], CS110[2], CS143[4], CS149[15], CS161[16], CS166[3], CS205L[17], CS224N[5], CS243[18] の 11 科目を対象に実験を行った。

システムは、**array**, **stack**, **heap**, **memory**, **queue**, **pointer**, **tree**, **hash** という、CS で用いられる特徴的な用語 8 つについてスコアを求めた。C-TF-IDF(通常の C-TF-IDF), 正規化した C-TF-IDF, 正規化と重み付けを合わせた C-TF-IDF(重み付けした C-TF-IDF) の 3 種類のスコアリング手法を比較する。スコアに対して 2 種類の実験を行った。

筆者による実験 対象となるページを網羅的に確認し、スコアリングの上位が妥当なものか検証する実験である。この実験では、科目を独立に扱い、それぞれの単語ごとに筆者が「将来のページとして出てきて欲しい」という基準でランキングを作成した。3 種類のスコアリング手法における 3 位と筆者が作成した上位 3 位以内の結果を評価の対象とする。実際の対象は 11 科目合計で 69 件、ページ数は合計で 5436 ページであった。

被験者による実験 スコアリング上位のページを対象として、順位付けが妥当か筆者以外が判断する実験である。この実験では、科目別の結果を統合し、3 種類のスコアリング手法における上位 5 位のページの結果から重複を手動で削除し、被験者にランキングの作成を依頼した、基準となるページに対して、「一緒に表示されて嬉しいページ」順にランキングを作成してもらった。ランキングの上位 5 位の結果を評価の対象とする。対象となったページは合計で 81 ページである。

4.1 実験の評価指標

本研究では評価の指標として Q-measure を用いる。Q-measure は酒井 [19] が提案している情報検索システムの評価に用いられる手法の 1 つである。この手法では、正解となるランキングを用意し、1 位から順に得点を与える。そして、システムが求めたランキングで、正解のランキングの順位のものが出現するタイミングでブレンド比という値を計算する。ブレンド比は、得点の値がどのような場合でも理想的な順序とシステムが出した順序の比を求めるために用いる。ブレンド比 $BR(r)$ は、以下の定義を用いて、式 (16) と定義できる。ここで、 r はランキング第 r 位、 $cg(r)$ は第 r 位までにシステムが受け取った得点の合計、 $count(r)$ は第 r 位までの得点を受け取った回数、 $cg_I(r)$ は第 r 位までの理想的な得点の合計となる。

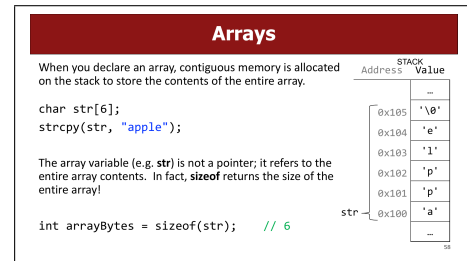


図 2 学習中に **array** を選択したと仮定 (CS107[14] より引用)

$$BR(r) = \frac{cg(r) + count(r)}{cg_I(r) + r} \quad (16)$$

Q-measure は、「各正解が検索された時点でのブレンド比を、全正解で平均したもの」[19] としている。式 (17) は、Q-measure を求める式である。 L は検索するデータのサイズ、 R は r までの正解の数、 I は得点の有無のフラグで 1 か 0 となる。

$$Q\text{-measure} = \frac{\sum_{r=1}^L I(r)BR(r)}{R} \quad (17)$$

この手法を用いることで、正解との順位も含め近いランキングの評価が高くなる。本研究では、正解となるランキングと得点について、筆者が作成したランキングでは第 1 位から順に 3 点、2 点、1 点とし、被験者が作成したランキングでは得点を第 1 位から順に第 5 位までを 5 点、4 点、3 点、2 点、1 点として Q-measure を求める。

4.2 筆者による実験の結果

表 1 は、CS166[3] において、図 2 の **array** に注目したと仮定した時の結果である。Q-measure の値を表 2 に示す。#16 の P.77 を図 3 に、#17 の P.33 の結果を図 4 に示す。図 3 は #16 の最後のページで次回実施する内容について述べているが、こういったページよりも詳細を述べたページを示す方が有効であると考えられる。図 4 は文字列探索のアルゴリズムである Suffix Arrays について取り上げ、探索の対象の文字列が「banana」である場合の例である。このような詳細を述べたページを示すことは有効であると考えられる。Q-measure の結果を見ると、重み付けした C-TF-IDF は、筆者が想定するページを選ぶことができているといえる。

表 3 は、図 5 を学習中に **memory** に注目したと仮定した時の CS107[14] のランキングである。図 6 は、重み付けした C-TF-IDF と筆者が作成したランキング内で第 1 位となったページである。学習中と仮定したページでは、「CS107 でより深く説明される」とされており、実際に関連するページを示した例である。このようにして発展的な内容を直接示すことで、学生は **memory** に対する学習内容を整理することができる。また、図 7 は、メモリークについて説明しているページである。このページは筆者がランキング作成時には考慮していなかったが、システムは

表 1 CS166[3] における array のランキング

	通常の C-TF-IDF	正規化した C-TF-IDF	重み付けした C-TF-IDF	筆者作成のランキング
	Page	Page	Page	Page
1 位	#16 p.77	#16 p.77	#04 p.105	#17 p.33
2 位	#10 p.165	#10 p.166	#18 p.4	#04 p.105
3 位	#04 p.106	#17 p.33	#16 p.77	#10 p.118

表 2 表 1 の Q-measure の値

	Q-measure
通常の C-TF-IDF	0.1481
正規化した C-TF-IDF	0.1481
重み付けした C-TF-IDF	0.5039


Next Time

- **Suffix Arrays**
 - A space-efficient alternative to suffix trees.
- **LCP Arrays**
 - Implicitly capturing suffix tree structure.

図 3 #16 の P.77(CS166[3] より引用)

Suffix Arrays

- A **suffix array** for a string T is a sorted array of the suffixes of the string T .
- Suffix arrays distill out just the first component of suffix trees: they store suffixes in sorted order.



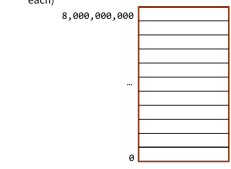
\$
AS
ABANANABANDANAS
ABANDANAS
ANAS
ANABANDANAS
ANANABANDANAS
ANDANAS
BANANABANDANAS
BANDANAS
DANAS
NAS
NABANDANAS
NANABANDANAS
NDANAS

ABANANABANDANAS

図 4 #17 の P.33(CS166[3] より引用)

How does this look in memory?
A little background...

- A computer's memory is like a giant Vector/array, and like a Vector, we start counting at index 0.
- We typically draw memory vertically (rather than horizontally like a Vector), with index 0 at the bottom.
- A typical laptop's memory has billions of these indexed slots (one byte each)



* Take CS107 to learn much more!!

Stanford University

図 5 学習中に memory を選択したと仮定 (CS106B[1] より引用)

発見したページである。こうしたメモリークに関する内容を、初学者が意識することで、初めて実装する際には注意を払うことができる。加えて、学生が実装に慣れてきた頃にこのページを示すことで、自らの実装を見直すことが

Memory

- Memory is a big array of bytes.
- Each byte has a unique numeric index that is commonly written in hexadecimal.
- A pointer stores one of these memory addresses.

Address	Value
...	...
0x105	'\0'
0x104	'e'
0x103	'l'
0x102	'p'
0x101	'p'
0x100	'a'
...	...

21

図 6 CS107[14] の#5 の P.21

Memory Leaks

- A memory leak is when you allocate memory on the heap, but do not free it.
- Your program should be responsible for cleaning up any memory it allocates but no longer needs.
- If you never free any memory and allocate an extremely large amount, you may run out of memory in the heap!

However, memory leaks rarely (if ever) cause crashes.

- We recommend not to worry about freeing memory until your program is written. Then, go back and free memory as appropriate.
- Valgrind is a very helpful tool for finding memory leaks!


68

図 7 CS107[14] の#7 の P.68

Queues

queue: First-In, First-Out ("FIFO")

- Elements stored in order they were added
- Can add only to the back, can only examine/remove frontmost element



queue operations

- enqueue: Add an element to the back
- dequeue: Remove the front element
- peek: Examine the front element

Stanford University

図 8 学生が queue を選択したと仮定 (CS106B[1])

できる。表 4 の Q-measure の値は重み付けした C-TF-IDF が最も高くなった。特に第 1 位が一致しているため、最も重要と考えられるページを発見できているといえる。

4.3 被験者による実験の結果

図 8 の中で基礎科目を学習中の学生が queue を選んだと仮定したランキングを表 5 に示す。被験者が作成したランキングに基づく Q-measure の値と平均を表 6 に示す。被験者全体で 8 名中 7 名が、第 5 位以内に選んだ、CS166 の #4 の P.193 を図 9 に示す。このページは、Two-Stack Queue の計算量について述べているページである。CS166 の #4 の P.33 は図 10 である。CS166 の #4 の P.13 や P.33 は、パラパラ漫画ように連続したページの一部である。これらのページは一つにまとめ、アニメーションのように示すことで、学生にわかりやすい提示が可能になると考え

表 3 CS107 における memory のランキング

	通常の C-TF-IDF	正規化のみの C-TF-IDF	重み付けした C-TF-IDF	筆者作成のランキング
1 位	#05 p.93	#11 p.10	#05 p.21	#05 p.21
2 位	#07 p.68	#02 p.46	#11 p.10	#05 p.72
3 位	#02 p.46	#07 p.52	#07 p.68	#02 p.46

表 4 表 3 の Q-measure の値

	Q-measure
通常の C-TF-IDF	0.0741
正規化した C-TF-IDF	0.1693
重み付けした C-TF-IDF	0.6720

表 5 queue のランキング

	通常の C-TF-IDF	正規化のみの C-TF-IDF	重み付けした C-TF-IDF
1 位	CS166 #04 p.13	CS166 #04 p.13	CS166 #04 p.33
2 位	CS149 #05 p.18	CS149 #05 p.37	CS166 #04 p.103
3 位	CS106L #04 p.05	CS149 #05 p.18	CS149 #13 p.34
4 位	CS149 #05 p.11	CS149 #05 p.35	CS166 #04 p.193
5 位	CS149 #05 p.37	CS149 #05 p.11	CS166 #05 p.16

られる。一方で、こうした説明がある科目はページ数が多く、他の科目よりも値が高くなる傾向が見られた。そのため、科目間のページ数についても正規化を行う必要があると考えられる。queue についてランキングの作成に関するコメントの一部を表 7 に示す。学習中と仮定した図 8 では、queue の概念的なイメージを示したため、より具体的な実装を行う面を重視したコメントが多い。Q-measure の値が最も高くなった、重み付けした C-TF-IDF は、被験者 H が望む「先入れ先出しの動き」という queue の動作に関わる部分を示すことに成功しているといえる。一方、被験者 E に対しては通常の C-TF-IDF のほうが有効であった。おそらく被験者 E は、表 7 のコメントから、queue 自体がどういったものか知りたいと考え、そうしたページを上位に選んだと考えられる。これらから、重み付けした C-TF-IDF は動作について細かく説明したものを取得したい場合に有効で、通常の C-TF-IDF は単語自体の説明しているページを取得したい場合に有効である。被験者実験では memory 以外の全てで、重み付けした C-TF-IDF のスコアが高くなった。そのため、重み付けした C-TF-IDF は被験者が想定するページを反映していると考えられる。

4.4 全体の考察

今回、結果の中に含まれていないが、被験者実験の中で CS205L や CS243 についてはランキングに入ることは

表 6 被験者ごとの Q-measure の値

被験者	通常の C-TF-IDF	正規化のみの C-TF-IDF	重み付けした C-TF-IDF
A	0.2105	0.180	0.3345
B	0.3160	0.1877	0.3752
C	0.040	0.2527	0.6476
D	0.0422	0.020	0.4989
E	0.5527	0.3878	0.0
F	0.3264	0.2866	0.2488
G	0.4197	0.4745	0.0422
H	0.2778	0.3631	0.7336
平均	0.282	0.282	0.368

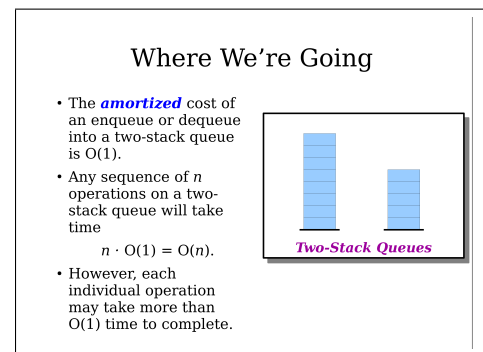


図 9 CS166[3] の #4 の P.193

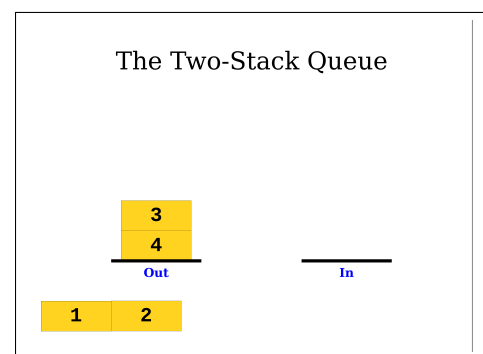


図 10 CS166[3] の #4 の P.33

表 7 queue で得られたコメントの一部

被験者	コメント
D	関連性がありそうなものとスライド 1 枚で内容が分かりやすそうなものを優先して選んだから
E	なんの説明をするかがほしいと思ったから
H	先入れ先出しの動きが知りたいから

なかった。これらの科目では、今回対象とした単語があまり出現しなかった。単語が基礎科目の内容に寄っていたこ

とが考えられるため、今後別の単語でも調査が必要である。

本研究では単語を直接結びつけていた。一方、学生が注目しているページの他の単語は考慮していない。ページの特徴を重視する場合、共起行列やコサイン類似度を用いてページの特徴同士を比較することが考えられる。これらの手法と組み合わせることで、図 10 のようなパラパラ漫画のページを発見してまとめることも可能となる。

5. まとめ

本研究では、自習中の学生の注目箇所に基づいて、科目を超えて過去や未来のページを発見するためのスコアリング手法の提案を行った。スコアリングには TF-IDF を用い、授業資料が持つ特徴に合わせて、時系列と重み付け、正規化を用いてスコアを補正した。評価用のシステムを作成し、Stanford University の授業資料を用いて、提案手法のスコアリングに対する評価実験を行った。筆者がランキングを作成した実験と、被験者がランキングを作成した実験の 2 種類を実施した。双方の実験で通常の C-TF-IDF、正規化のみの C-TF-IDF、重み付けした C-TF-IDF の 3 つのスコアの順位を Q-measure を用いて比較した。実験から、重み付け C-TF-IDF の Q-measure の値は重み付けしないものと比べて高い傾向となった。これらのことから、重み付けした C-TF-IDF は有効なページを示すことができることが明らかになった。

今後の課題として、過去と未来でスコアリングの手法を変えることが考えられる。現在の手法は未来の科目を示すことに適している。過去の科目に示すために、学習中のページから遡る形でスコアリングを行うことが考えられる。また、第 1 章で示したステップ 1 やステップ 3 についても検討が必要である。ステップ 1 は、学生があるページにたどりついた時点で全ての単語を取得するようにすることで良いと考えている。ステップ 3 は、実験では筆者が第 3 位までを科目別に、被験者には科目別のスコアを合わせて第 5 位までとした。システムが算出した結果を考慮すると第 5 位までを科目別で示すことが効果的であると考えられるため、実際にどのようなユーザインタフェースで提示するかを含め、検討を行う。

参考文献

[1] Lee, C. B., Zelensk, J. and Kishnani, N. S.: CS106B Programming Abstractions, <https://web.stanford.edu/class/archive/cs/cs106b/cs106b.1222/> ((Retrieved Dec. 22 2021)).

[2] Cain, J.: CS110 Principles of Computer Systems, <https://web.stanford.edu/class/archive/cs/cs110/cs110.1222/> ((Retrieved Dec. 22 2021)).

[3] Schwarz, K.: CS166 Data Structures, <http://web.stanford.edu/class/cs166/> ((Retrieved Dec. 22 2021)).

[4] Kjolstad, F. and Aiken, A.: CS143 Compilers, <https://web.stanford.edu/class/cs143/> ((Retrieved Dec. 22 2021)).

[5] Manning, C.: CS224N Natural Language Processing with Deep Learning, <http://web.stanford.edu/class/cs224n/> ((Retrieved Dec. 22 2021)).

[6] Goncharow, A., McQuaigue, M., Saule, E., Subramanian, K., Payton, J. and Goolkasian, P.: Mapping Materials to Curriculum Standards for Design, Alignment, Audit, and Search, *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, SIGCSE '21*, New York, NY, USA, pp. 295–301 (2021).

[7] 山本雄介, 峰松 翼, 長沼祥太郎, 谷口雄太, 大久保文哉, 島田敬士: 科目の関連性情報を付加したカリキュラム情報閲覧システムの開発, 情報処理学会研究報告教育学習支援情報システム研究会 (CLE), Vol. 2020-CLE-35, No. 9, pp. 1–8 (2021).

[8] Sajjacholapunt, P. and Joy, M.: Analysing Features of Lecture Slides and past Exam Paper Materials, *Proceedings of the 7th International Conference on Computer Supported Education- Volume 1*, pp. 169–176 (2015).

[9] 桐原牧紀, 王 元元, 河合由起子, 角谷和俊: e-Learning における講義コンテンツの階層構造に基づくスライド推薦方式の提案, 第 13 回データ工学と情報マネジメントに関するフォーラム (DEIM Forum 2021), pp. 1–6 (2021).

[10] Sato, S., Hayashi, H., Maki, N. and Inoguchi, M.: Development of Automatic Keyword Extraction System from Digitally Accumulated Newspaper Articles on Disasters, *2nd International Conference on Urban Disaster Reduction Proceedings*, pp. 1–6 (2007).

[11] Paik, J. H.: A Novel TF-IDF Weighting Scheme for Effective Ranking, *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '13*, New York, NY, USA, p. 343–352 (online), DOI: 10.1145/2484028.2484070 (2013).

[12] Imbe, T. and Maruyama, K.: *Finding the Noteworthy Learning Material about a Keyword through Courses Already Learned and to Be Learned*, p. 1268 (online), available from (<https://doi.org/10.1145/3408877.3439596>), Association for Computing Machinery (2021).

[13] Cerkenvenik, F. and Edamadaka, S.: CS106L Standard C++ Programming, <http://web.stanford.edu/class/cs106l/index.html> ((Retrieved Dec. 22 2021)).

[14] Troccoll, N.: CS107 Computer Organization & Systems, <https://web.stanford.edu/class/archive/cs/cs107/cs107.1216/> ((Retrieved Dec. 22 2021)).

[15] Fatahallan, K. and Olukotun, K.: CS149 Parallel Computing, <http://35.227.169.186/cs149/fall119/> ((Retrieved Dec. 22 2021)).

[16] Shi, K.: CS161 Design and Analysis of Algorithms, <https://web.stanford.edu/class/cs161/> ((Retrieved Dec. 22 2021)).

[17] Fedkiw, R.: CS205L Continuous Mathematical Methods with an Emphasis on Machine Learning, <http://web.stanford.edu/class/cs205l/> ((Retrieved Dec. 22 2021)).

[18] Lam, M. S.: CS243 Program Analysis and Optimization, <https://suif.stanford.edu/~courses/cs243/> ((Retrieved Dec. 22 2021)).

[19] 酒井哲也: よりよい検索システム実現のために: 正解の良し悪しを考慮した情報検索評価の動向, 情報処理, Vol. 47, No. 2, pp. 147–158 (2006).