情報処理学会論文誌 Vol.61 No.4 930–944 (Apr. 2020)

ソフトウェアドキュメンテーションのための クラスタ内文書ランキング

溝渕 裕司[1,a),b)] 吉田 和樹[2,c)]

受付日 2019年8月1日, 採録日 2020年1月16日

概要：ソフトウェアドキュメンテーションとは，ソフトウェアに付随する文書（ソフトウェア文書）を作成することであり，各ステークホルダーが各々にとって関心のある情報を文書化することである．ソフトウェア文書に記載されていない，もしくは最新化されていない情報は，クラウド文書で議論されていることが多く，広く利用されている．クラウド文書で頻出する問合せ（FAQs）は，ソフトウェア文書へ反映することによってクラウド文書への問合せを減らせ，作業の効率化を期待できる．しかし，ソフトウェア文書への反映には事前に FAQs の読解が必要であり，作業リソースが限られるなか，すべての文書を読解することは不可能である．本稿では，膨大な FAQ の読解を効率的かつ効果的に実施するために，優先的に読むべき問合せをランキングするクラスタ内文書ランキング手法を検討・評価した．具体的には，文書クラスタごとに単語を重み付けする手法，ICDF を開発し，これをクラスタ内文書ランキングに適用することで A-NDCG 0.703 を達成した．

キーワード：ソフトウェアドキュメンテーション，FAQ，クラスタリング，文書ランキング，単語の重み付け，機械学習

Intra-Cluster Document Ranking for Software Documentation

Yuji Mizobuchi[1,a),b)] Kazuki Yoshida[2,c)]

Received: August 1, 2019, Accepted: January 16, 2020

Abstract: Software documentation is the creation of software documents according to the perspective of each software stakeholder. The information that is not documented or updated in software documents is often discussed in crowd documents, so crowd documents are also widely used. Frequently asked questions (i.e. FAQs) in the crowd documents reveal that there are many inquiries about the same question, so it is expected that the number of inquiries can be reduced by reflecting FAQs to the software documents. However, it is impossible to read all the FAQs since the volume of FAQs is sometimes too large and resources are limited, though it is required to read FAQs beforehand. In this paper, we examined and evaluated an Intra-Cluster Document Ranking that can rank documents within a specific question cluster. And this enables us to read the questions in an appropriate order. Specifically, we developed ICDF, a method of weighting words based on document clusters and applied it to the Intra-Cluster Document Ranking. Then we finally achieved A-NDCG 0.703.

Keywords: software documentation, FAQ, clustering, document ranking, term weighting, machine learning

[1] 株式会社富士通研究所
FUJITSU LABORATORIES LTD., Kawasaki, Kanagawa 211–8588, Japan
[2] 国立情報学研究所
National Institute of Informatics, Chiyoda, Tokyo 101–8430, Japan
[a)] mizobuchi.yuji@fujitsu.com
[b)] yuji.mizobuchi@gmail.com
[c)] k-yoshida@nii.ac.jp

1. はじめに

ソフトウェアドキュメンテーションとは，ソフトウェア文書（Software Documents）を作成することであり，ソフトウェア文書とは，ソフトウェアの各ステークホルダーのそれぞれの立場に応じた関心事が記載された文書である．

© 2020 Information Processing Society of Japan

たとえば，ソフトウェア開発者はソフトウェアの開発・保守工程で設計情報や過去の変更履歴に関心があり，要件定義書，設計書，テスト仕様書などを文書化する．また，ソフトウェア開発者は，ソフトウェア利用者にその利用方法を伝えるために，API ドキュメント，利用手引きなどを文書化する．なお，文書とは参照されることを前提として記録されるものと，その内容である．たとえば，紙に文字を記録したものがある．また，紙のような媒体に文字などの情報を記録することを文書化という．

一般にソフトウェア文書はソフトウェア管理母体が公認した文書であり，API ドキュメントや利用手引きはソフトウェア利用者が第一に参照する文書である．したがって，正確かつ網羅的に整備されていることが望ましい [1], [2]．しかし，ソフトウェア文書はつねに陳腐化にさらされており，信頼できる情報源として利用され続けるためには，新たに発見された情報を継続的に反映していく必要がある．

一方，記載されていない，もしくは最新化されていない情報がソフトウェア文書にある場合には，クラウド文書（Crowd Documents）と呼ばれるものが利用されている [3], [4], [5]．クラウド文書とは，ソフトウェアの運用・保守工程でソフトウェア管理母体に限定しないさまざまなステークホルダーによって利用される文書である．たとえば，OSS コミュニティフォーラムでのスレッドや E-mail（特にメーリングリスト），Q&A サイトの問合せやその回答，また Wikipedia [6] などのクラウドソーシングによる文書があり，新たに見つかったバグなど想定外の事象について議論されている．このようにクラウド文書はさまざまなステークホルダーによる情報であるため，信頼性，整合性，精度，冗長性などに課題があるものの，多数のステークホルダーによる迅速な情報共有が可能であるため，有用な情報として利用されている．

これら異なる形態の文書は補完し合う関係にあるが，可能な限り一元化を図ること，すなわちクラウド文書からソフトウェア文書へ情報反映されることが望ましい．なかでも類似するクラウド文書のクラスタは，ソフトウェア文書への同じ問合せを多数失敗した結果であり，早急な情報反映によって不要な問合せを削減できる．

本稿では，上記クラウド文書のうち，Q&A サイトの投稿に焦点を当て，頻出する類似の問合せ（FAQs；Frequently Asked Questions）から FAQ（Frequently Asked Question）を作成し，それをソフトウェア文書へ反映する作業（以後，情報反映作業と呼ぶ）の効率化を検討する．なお，FAQ とは FAQs を整理・抜粋・要約・推敲した文書であり，FAQs から FAQ を作成することを FAQ 作成もしくは FAQ 化，その作業を FAQ 作成作業もしくは FAQ 化作業と呼ぶ．

情報反映作業は，概して，作業 1：FAQs の選出，作業 2：FAQ の作成，作業 3：ソフトウェア文書への FAQ の反映から構成され，問合せが膨大になると多大なコストを要する．たとえば，2018 年 9 月時点の Stack Overflow Dump Data [7] からは，問合せ数が 3 件以上の FAQs は 35,352 個あり，そのうち最も大きいものは 195 件から構成されていた．とりわけ作業 2 では，FAQs の内容把握が必要であり，作業者の読解量に限りがあるなか，いかに重要な文書を選択し効率的に読解していくかがポイントとなる．

本稿では，限られたリソースで FAQ 作成を実現するべく，FAQs から読むべき優先度の高い問合せをランキングする方式（以後，クラスタ内文書ランキングと呼ぶ）を検討・評価する．クラスタ内文書ランキングは通常の文書ランキングと異なり，(1) クエリを使わない点，(2) 類似する文書の集まり（以後，文書クラスタもしくはクラスタと呼ぶ）をランキング対象とする点が特徴といえる．特に (2) の特徴は，ランキングに使える特徴量が少ないことを意味し，不特定多数の文書を対象とする一般の文書ランキングより高精度化が難しい．なお，著者らが調査した限りでは，クラスタ内文書ランキングに関する報告は過去になく，本稿が初めての取り組みである．

クラスタ内文書ランキングでは，文書クラスタが所与であることを前提とするため，これを活かした単語の重み付け手法 ICDF（Inverse Complemental Document Frequency）を開発した．これは従来から自然言語処理分野で多用される TF-IDF（Term Frequency Inverse Document Frequency）の考え方を参考にしたもので，クラスタ内の単語の出現頻度を考慮した単語の重み付け手法である．さらに ICDF を応用して文書の重要度の算出手法 CTI2（Cumulative TF-IDF-ICDF）および特徴量選択手法 TI2-Filter（TF-IDF-ICDF Filter）を開発した．とりわけ TI2-Filter による特徴量削減を行った多層パーセプトロンによる実験では，クラスタ内文書ランキングの指標である A-NDCG で 0.703 を達成した．

なお，本稿は，国立情報学研究所の社会人向け教育プログラムであるトップエスイー [8] のソフトウェア開発実践演習での活動をもとに執筆した．

2. 情報反映作業の課題と支援方法

本節では，FAQs をソフトウェア文書に反映する情報反映作業とそのうち本稿が注目した支援対象作業およびその支援方法について説明する．

2.1 情報反映作業の流れ

本節では情報反映作業の流れを**図 1** と合わせて説明する．情報反映作業の流れは次のとおりである．

作業 1：FAQs の選出

概要：不特定多数の問合せから人手またはクラスタリングよって FAQs を特定し，何らかの基準に基づいてソフトウェア文書へ情報反映が必要な FAQs を選出する．たとえば，ある一定以上の頻度で発生して

© 2020 Information Processing Society of Japan

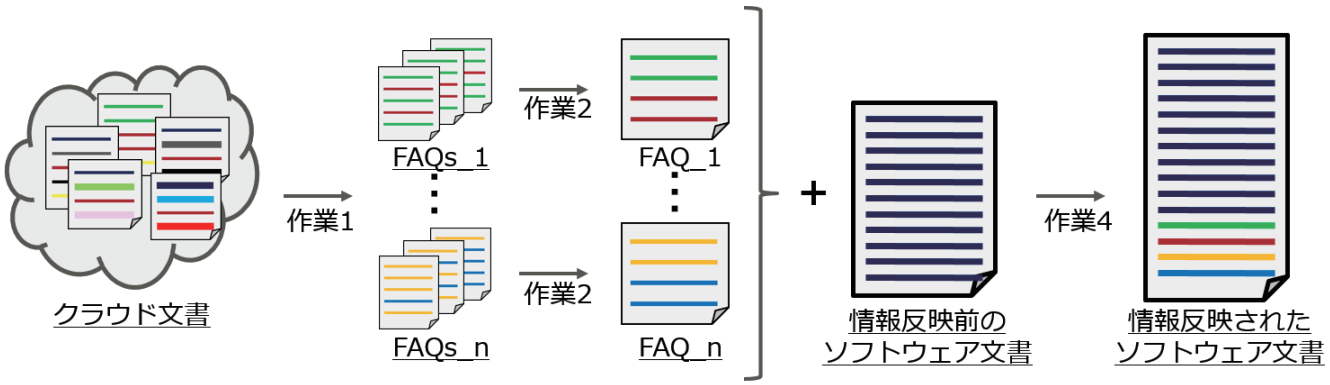

図 1 情報反映作業の流れ

Fig. 1 Workflow of updating the software document with crowd documents.

いたり，優先度の高い内容の FAQs であったりする場合，FAQ 化の対象となる．また，ソフトウェア文書に反映済の FAQs は除外し，残りを反映対象として残す．

入力：不特定多数の問合せ

出力：FAQ 化が必要と判断された FAQs

作業 2：FAQ の作成

概要：処理 1 で選出された FAQs の記載内容を過不足なく整理・抜粋・要約・推敲し，FAQ を作成する．本作業は以下の作業から構成される．

作業 2-1：FAQs の読解

作業 2-2：内容の網羅性・重要度を考慮した整理・抜粋

作業 2-3：整理・抜粋した内容の要約・推敲および可読性の高い文章の作成

入力：FAQs

出力：FAQ

作業 3：ソフトウェア文書への FAQ の反映

概要：処理 2 で作成された FAQ をソフトウェア文書へ反映する．

入力：FAQ およびソフトウェア文書

出力：情報反映されたソフトウェア文書

2.2 FAQs 読解の課題と支援方法

本節では，情報反映作業のうち本稿で注目する作業 2-1 の FAQs の読解作業における課題およびその支援方法について説明する．

FAQ には，可能な限り FAQs に記載されている内容を漏れなく盛り込むことが望ましい．そのため，FAQ 作成に必要なリソース（時間や労働力など）を際限なく使えるのであれば FAQs のすべての文書を読解することが可能となり，上記要件は達成できる．しかし，実際には FAQs が大規模化してくると，それに見合う潤沢なリソースを確保することは難しく，いかに FAQ 化に有用な問合せを絞り込むかが課題となる．

ある問合せが FAQ 化に有用とは，それをソフトウェア文書へ反映したのち，その情報が参考になる度合い（以後，参考度と呼ぶ）による，といえる．読まれる機会の少ない情報は，ソフトウェア文書へ反映してもそれによる効果が小さいためである．一方，ソフトウェア文書で参考度の高い情報は，FAQs の時点で参考度が高い場合が多い．たとえば，Stack Overflow では，得票数の多い問合せは多くの利用者から参考になると評価されており，このような文書を優先的にソフトウェア文書へ反映していくことが望ましい．

以上をふまえ，本稿では，FAQs の問合せを参考度に応じてランキングすることで，読解すべき FAQs の絞り込みを可能にする．

3. 関連研究

本節では関連研究について説明する．

従来からソフトウェアに関する議論の場として，Eclipse Community Forums [9] や Bugzilla [10] などの OSS コミュニティフォーラム，また，Stack Overflow [11] などのナレッジコミュニティがあった．これらを介してさまざまな質問や回答が行われてきており，多くのクラウド文書が蓄積されている．蓄積された情報を効率的かつ効果的に活用するため，情報検索，FAQ 作成支援，投稿内容の分析・評価手法に関する研究が行われてきた．

3.1 クラウド文書に対する情報検索

本節では，クラウド文書に対する情報検索に関する関連研究について説明する．

従来からさまざまなクラウド文書に対して情報検索技術が研究されてきた．たとえば，OSS コミュニティフォーラムで蓄積された文書に対する情報検索に関しては，[12], [13]，ナレッジコミュニティで蓄積された文書に対しては，[14], [15], [16] がある．いずれも文書内の項目ごとの単語の共起性とその重みに基づいて文書間の類似度を評価するモデルを用意し，そのモデルを用いて新規の問合

© 2020 Information Processing Society of Japan

せと既存の文書を類似度に基づいてランキングする．

また，プログラム開発・保守支援を目的に，Stack Overflow の Q&A のうち実装方法に関するものに絞って推薦する方法 [17]，Stack Overflow の Q&A に対して文書検索と要約を組み合わせて冗長・無関係な検索結果を整理し，高精度化を実現する方法 [18]，クラッシュ情報から関連する Stack Overflow の Q&A 内のコード断片を検索・列挙し，それら複数のコード断片から修正コードを生成・推薦する方法 [19] がある．

いずれの研究も検索クエリが与えられることを前提に既存の文書をランキングする方法であり，検索クエリを用いないクラスタ内文書ランキングに適用することはできない．

3.2 クラウド文書に対する FAQ 作成支援

本節では，クラウド文書に対する FAQ 作成支援に関する関連研究について説明する．

FAQ 作成作業の 1 つにクラウド文書をクラスタリングした結果からの FAQs の特定および FAQ 化の候補の選定がある（2.1 節の作業 1 に相当）．当該作業の効率化を目的にさまざまなデータを用いた研究が報告されてきた．たとえば，OSS コミュニティフォーラムの問合せで行った研究 [20], [21]，ヘルプデスクや IT システムの問合せで行った研究 [22], [23] がある．

そのほかにプログラミング開発時の操作ログ，API ドキュメントや Web ページの閲覧履歴を用いて，API ドキュメントに FAQs を関連付けする研究 [24]，オンライン学習サービスのディスカッションボードの Q&A を参照回数や投票数に基づいて FAQ に活用するか判別する技術の研究 [25] がある．

いずれもクラウド文書からの FAQs の選出作業を対象にしているが，FAQ 作成時の読解作業を対象としていない．

3.3 クラウド文書の分析・評価

本節では，クラウド文書の内容分析・評価に関する関連研究を説明する．

Ponzanelli ら [26] は，Stack Overflow に投稿される質問・回答の品質を，投稿内容のテキストメトリクスや読解力メトリクスから測定するモデルを提案した．品質の尺度には投稿内容へ付与される得票数（Stack Overflow の Vote）を用いている．Xia ら [27] も同様に投稿内容から将来削除される投稿かどうかを予測するモデルを提案した．また，Yao ら [28] は，質問と回答の獲得投票数に相関があることを発見し，品質予測に活用した．いずれの研究も対象文書群に対して特段の選別は行っていない．

一方，本稿が対象とする類似する文書のクラスタを対象とした場合，文書間の差異として使える特徴量が少ない．したがって，相対的に品質の予測精度を出しにくい問題といえる．**図 2** は Stack Overflow の類似（Duplicate）タグ

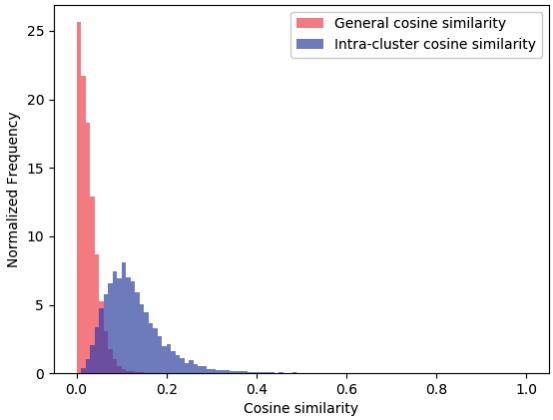

図 2 クラスタ内の問合せ間とランダムに選ばれた問合せ間のコサイン類似度の分布

Fig. 2 Distribution of cosine similarities between inter-cluster questions and general questions.

でつながる類似の問合せクラスタを集め，クラスタ内の問合せのペアとランダムにサンプルリングした問合せのペアでのコサイン類似度の分布を示している．なお，コサイン類似度に用いたベクトルは問合せのタイトル，本文，タグに現れる単語の有無を表したもの（BOW；Bag of Words）である．これより，同一クラスタ内の問合せ間のコサイン類似度はランダムにサンプリングした場合に比べて高いことが分かる．本稿では，クラスタ内文書ランキングを想定した単語の重み付け手法および文書メトリクスを開発し，精度を改善した．

その他，クラウド文書の分析技術として，バグレポートに対する修復作業に必要な情報の記載有無を判別する方法 [29] がある．具体的には，(1) バグの現象，(2) その再現方法，(3) 期待される挙動，をレポート内で特定する．また，E-mail の内容理解を目的に E-mail に込められた意図を判別する方法 [30] がある．FAQ 作成においても特定の文書の絞り込みや文書内の情報の特定を行う際に必要な技術であり，本稿の補完技術として活用できる．

4. 準備

本節では本稿で必要となる予備知識を説明する．

4.1 TF-IDF

従来から自然言語処理の単語重み付け手法として，TF-IDF [31], [32] がある．情報検索やテキストマイニングの調査によれば，TF-IDF は単語の重み付け手法として最も使われてきた手法であり [33], [34]，本稿も参考にした．

TF-IDF は TF（Term Frequency；単語の出現頻度）と IDF（Inverse Document Frequency；逆文書頻度）の積で求められ，以下の式で表される．

$$\text{tf-idf}_{i,j} = \text{tf}_{i,j} \cdot \text{idf}_i$$

© 2020 Information Processing Society of Japan

$$\mathrm{tf}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$
$$\mathrm{idf}_i = \log \frac{|D|}{|\{d : d \ni t_i\}|}$$

ここで $n_{i,j}$ は文書 d_j における単語 t_i の出現回数の和，$|D|$ は総文書数，$|\{d : d \ni t_i\}|$ は単語 t_i を含む文書数である．

本稿では，TF-IDF の考え方を参考に文書クラスタを考慮した単語の重み付け手法，ICDF を開発した．

4.2 ランキングの評価指標

本節では，ランキングの評価指標で代表的な評価指標を説明し，クラスタ内文書ランキングの評価指標のベースに NDCG [35] を用いた理由を説明する．

一般に文書ランキングでは，ユーザの関心事を表すクエリを発行して不特定多数の文書の順位付けを行う [36]．この評価に必要な正解情報には，(1) 単にクエリと文書の関連の有無だけが与えられる場合と，(2) 関連度が与えられている場合があり，それぞれの場合で評価指標が異なる．本稿のクラスタ内文書ランキングの場合，事前に類似する文書，すなわち関連する文書の集まりに対してランキングを行うため，関連度を用いた評価が適用されることになる．

本節では，関連の有無で評価する MRR（Mean Reciprocal Rank）および MAP（Mean Average Precision）を説明し，次に関連度を用いた評価指標として，DCG（Discounted Cumulative Gain），NDCG（Normalized Discounted Cumulative Gain）を説明する．

4.2.1 MRR

本項では，MRR（Mean Reciprocal Rank）について説明をする．MRR の評価式は以下のとおりである．

$$\mathrm{MRR}_Q = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{k_q}$$

ここで Q は全クエリ，q は Q のうちのあるクエリであり，k_q はクエリ q に対して関連する文書が初めて現れた順位である．MRR は，概して文書ランキングによって如何に早く関連文書に到達できるかの指標であり，文書ランキングの利用において重要視される特性を指標化できている．しかし，上位の 1 件だけに注目し，その順位の逆数による評価であるため，いくつかの欠点も報告されている [37], [38]．加えて，クラスタ内文書ランキングではランキング対象の文書はすべて関連有りのものとなるため，本指標を適用した場合，評価値が必ず 1 となってしまい，評価指標として適さない．

4.2.2 MAP

本項では，MAP（Mean Average Precision）[36] について説明をする．MAP の評価式は以下のとおりである．

$$\mathrm{AveP}_q = \frac{\sum_{k=1}^{|D_{\mathrm{relevant}}|} \mathrm{Precision@k} \times \mathrm{rel}_{d_k,q}}{|D_{\mathrm{relevant}}|}$$

$$\mathrm{rel}_{d_k,q} = \begin{cases} 1 & (d_k \text{ is relevant to q}) \\ 0 & (\text{otherwise}) \end{cases}$$
$$\mathrm{MAP}_Q = \frac{1}{|Q|} \sum_{q \in Q} \mathrm{AveP}_q$$

ここで Precision@k はランキング結果の順位 k までの適合率であり，$\mathrm{rel}_{d_k,q}$ は，順位 k の文書 d_k とクエリ q の関連の有無（関連があれば 1，なければ 0）を表す．また，AveP_q はクエリ q に対するランキング結果に含まれる関連する文書の Precision@k の平均を求めたものであり，MAP_Q は全クエリ Q に含まれる各クエリ q の AveP_q の平均を求めたものになる．

MAP では，利用者が文書ランキングを上位からたどっていくとき，以下の振舞いを仮定している [38]．

仮定 1：関連する文書に到達した時点でたどるのを止める（つまり，関連しない文書ではたどるのを止めない）

仮定 2：いずれの関連する文書においても，たどるのを止める確率は同じ

しかし，現実には利用者はランキング結果の上位数件を読んだ時点でたどるのを止めることが多く，仮定 2 と合わない点が問題として報告されている．たとえば，PubMed [39] の利用ログを用いた調査 [40] やアイトラッキングシステムを用いた検索エンジンの調査 [41] によると，仮定 2 の確率分布は冪乗則に従うと報告されている．このように，文書ランキングの結果への罰則の与え方を上位から下位に従って調整することが望ましいことが分かる．

また，MRR と同様にクラスタ内文書ランキングで本指標を活用した場合，評価値が必ず 1 となってしまい，評価指標として適さない．

4.2.3 CG および DCG

本項から，関連度を用いた指標について説明する．関連度を用いた指標には，Järvelin ら [42] によって提案された CG（Cumulative Gain）および DCG（Discounted Cumulative Gain）があり，CG は下記の仮定 1，DCG は仮定 1 および仮定 2 を考慮した評価指標として提案された．

仮定 1：文書が持っているクエリに対する関連度が大きいほど有用である．

仮定 2：ランキングの上位に来る文書ほど有用である．

まず CG（Cumulative Gain）について説明する．CG の評価式は下記のとおりである．

$$\mathrm{CG}_{p,q} = \sum_{i=1}^{p} \mathrm{rel}_{d_i,q}$$

ここで $\mathrm{rel}_{d_i,q}$ は文書 d_i とクエリ q の関連度を表し，CG_p は p 件までの関連度の総和になる．したがって，本評価式は，上記仮定 1 を考慮した指標になっている．しかし，上位 p 件までの順序においては，どんな入れ替えを行っても結果に影響せず，仮定 2 を考慮した指標になっていない．

© 2020 Information Processing Society of Japan

さらに，Järvelin らは仮定 2 を考慮した指標として DCG（Discounted Cumulative Gain）を提案した．DCG の評価式は以下のとおりである．

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{\mathrm{rel}_i}{\log_2(\mathrm{i}+1)}$$

ここで $\mathrm{rel_i}$ はクエリと文書の関連度で，関数 $\log_2$ によって割り引かれた関連度の総和が求められる．これによりランキングの順位によって評価結果が影響を受け，仮定 2 を満たす評価指標となった．しかし，DCG は結果の長さや関連度の大きさに影響を受けて値の範囲がばらつくため，異なるランキング結果を直接比較できない．

4.2.4 NDCG

本項では，上記 DCG の問題点を改善した NDCG（Normalized Discounted Cumulative Gain）[35] について説明する．本指標は DCG の問題点を改善した指標として提案されたものであり，情報検索の指標として広く利用されている．NDCG の評価式は以下のとおりである．

$$\mathrm{NDCG}_p = \frac{\mathrm{DCG}_p}{\mathrm{DCG}_{\mathrm{Best}_p}}$$

この方法は予測ランキングの DCG（Discounted Cumulative Gain）を正解ランキングの DCG で正規化したもので，予測ランキングが正解ランキングにどの程度近いかを示すものである．これにより，DCG では直接比較ができなかった異なるクエリ間の評価を行えるようになった．

クラスタ内文書ランキングには，本指標を改良した A-NDCG を開発した．

5. 提案手法

本節では単語の重み付け方法である ICDF，およびその応用である文書の重要度の算出方法 CTI2 や特徴量選択手法 TI2-Filter について説明する．

5.1 ICDF（Inverse Complemental Document Frequency）

クラスタ内文書ランキングは事前に文書クラスタが所与であることを想定しているため，これを活かした単語の重み付け方法 ICDF を開発した．本手法では，TF-IDF による重み付けに加えて，次の 3 点を考慮することで高精度化を実現している．

- 全文書でまれに出現する単語は重要（すなわち IDF 値が高い）
- 同一クラスタ内で多くの文書で出現する単語は重要
- 同一クラスタ内でまれに出現する単語は重要ではない

文書全体で重要な単語が，あるクラスタ内で多く出現する場合，その単語はそのクラスタで重要といえ，一方，たとえ文書全体で重要な単語であっても，あるクラスタ内で出現する頻度が低い場合，そのクラスタ内で重要ではないとする．評価式は次のようになる．

$$\mathrm{tf\text{-}idf\text{-}icdf_{i,j,c}} = \mathrm{tf_{i,j}} \cdot \mathrm{idf_i} \cdot \mathrm{icdf_{i,c}}$$

$$\mathrm{tf_{i,j}} = \frac{\mathrm{n_{i,j}}}{\sum_{\mathrm{k}} \mathrm{n_{k,j}}}$$

$$\mathrm{idf_i} = \log \frac{|\mathrm{D_{all}}|}{|\{\mathrm{d} : \mathrm{d} \ni \mathrm{t_i} \wedge \mathrm{D_{all}} \ni \mathrm{d}\}|}$$

$$\mathrm{icdf_{i,c}} = \begin{cases} 0 & (|\{\mathrm{d} : \mathrm{d} \ni \mathrm{t_i} \wedge \mathrm{D_c} \ni \mathrm{d}\}| < \alpha) \\ \log \frac{|\mathrm{D_c}|}{|\mathrm{D_c}| - |\{\mathrm{d}:\mathrm{d} \ni \mathrm{t_i} \wedge \mathrm{D_c} \ni \mathrm{d}\}| + \alpha} & (\text{otherwise}) \end{cases}$$

ここで $\mathrm{n_{i,j}}$ は文書 $\mathrm{d_j}$ における単語 $\mathrm{t_i}$ の出現回数の和，$|\mathrm{D_{all}}|$ は総文書数，$|\mathrm{D_c}|$ はクラスタ c に属する文書数，$|\{\mathrm{d} : \mathrm{d} \ni \mathrm{t_i} \wedge \mathrm{D_{all}} \ni \mathrm{d}\}|$ は総文書中での単語 $\mathrm{t_i}$ を含む文書数，$|\{\mathrm{d} : \mathrm{d} \ni \mathrm{t_i} \wedge \mathrm{D_c} \ni \mathrm{d}\}|$ はクラスタ c での単語 $\mathrm{t_i}$ を含む文書数である．また，クラスタ内で低頻度な単語を除去する閾値として α を導入し，クラスタ内で α 回より低頻度の単語は重みが 0 となるようにした．

たとえば，**図 3** の word1 は 9 文書中 2 文書で現れているため IDF 値は高くなるが，Cluster1 では 3 文書中 1 文書でしか現れず，$\alpha = 1$ の場合 ICDF 値は 0 となる．結果，Cluster1 での word1 の重みは 0 となる．一方，word2 は 9 文書中 3 文書で現れているため IDF 値は word1 ほど高くならならない．しかし Cluster1 内ですべての文書で出現しており，結果的に Cluster1 では word1 より word2 のほうが重みが高くなる．また，Cluster1，Cluster2 で共通して現れる word3 は，Cluster1 では 3 文書中 2 文書に，Cluster2 内では 1 文書にのみ現れており，同じ単語でもクラスタごとに出現頻度が異なってくる．したがって，同じ単語でも異なるクラスタで ICDF 値が異なってくることが起こる．

5.2 CTI2（Cumulative TF-IDF-ICDF）

ICDF を応用した文書の重要度を測定する方法として，CTI2 を開発した．重要な文書ほど重要な単語を多く含んでいると考えられ，各文書に出現する単語の TF-IDF-ICDF 値の総和が高い文書は重要であると想定した．評価式は次のとおりである．

$$\mathrm{CTI2_{c,j}} = \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{tf\text{-}idf\text{-}icdf_{i,j,c}}$$

クラスタ c の文書 j の CTI2 は，文書 j 内の n 個の単語の TF-IDF-ICDF 値を求め，その総和を求める処理であることを示している．

5.3 TI2-Filter（TF-IDF-ICDF Filter）

機械学習では，入力特徴量が多い場合，事前に精度向上に寄与する特徴量の選択を行うことがある [43]．たとえば，入力特徴量に単語を用いる場合，低頻度もしくは高頻度に現れる単語は経験的に精度向上に寄与しないことが分かっ

© 2020 Information Processing Society of Japan

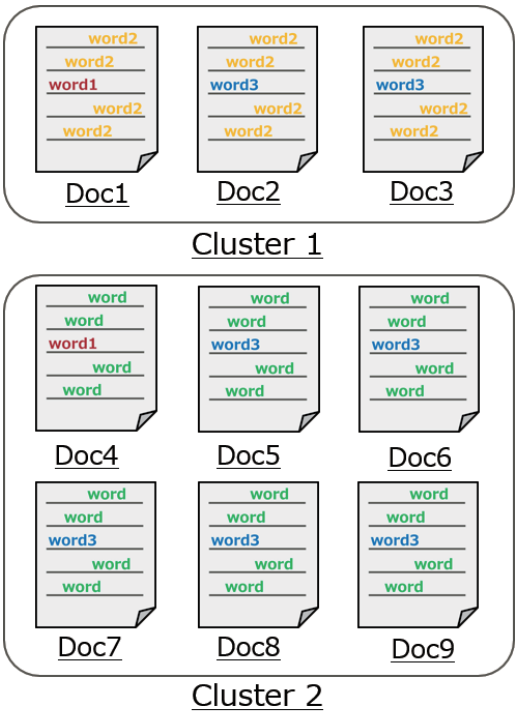

図 3 文書クラスタの例
Fig. 3 An example of document cluster.

ており，除外することが多い．本稿では学習ありクラスタ内文書ランキングを行う前に重要な単語を選択することで，精度向上を図った．単語の選択方法は次のとおりである．

$$\mathrm{Words}(\beta) = \{t_i \mid \exists \text{tf-idf-icdf}_{i,j,c} > \beta\}$$

これは，ある単語 t_i がクラスタ c の文書 j ごとに異なる TF-IDF-ICDF 値を持ち，そのうち 1 つでも閾値 β よりも大きい値を持つ場合，特徴量として残すことを意味する．

6. 実験方法

本節では実験方法について説明する．

6.1 実験データの準備

本節では実験データの作成方法を説明する．本稿では，2018 年 9 月 5 日発行の Stack Overflow の Dump データ [7] を用いた．Dump データのスキーマは [44] のとおりである．

6.1.1 FAQs の収集

本項では，Stack Overflow から FAQs を収集する方法を説明する．Stack Overflow では，サイト管理者や参加者が既出の内容の問合せに対して Duplicate というタグ [45] を付与できる．これは重複する問合せを減らし，真の回答への到達を容易にすることを目的としている．たとえば，**図 4** の例では，プログラミング言語 R [46] で文字列置換する方法に関して 3 つの文書が Duplicate タグで関連付けられている．

本稿では Duplicate タグで関連付けられていることを類似性の拠り所とし，このタグでつながる問合せの集まりを FAQs として活用する．また，このように Stack Overflow から集められた FAQs を問合せクラスタと呼ぶことにする．

なお，Duplicate タグは Dump データの PostLinks テーブルで管理されており，LinkTypeID=3 でつながれた問合せのペア（PostID と RelatedPostID）が Duplicate の関係であることを表す．問合せは Posts/PostsWithDeleted テーブルの PostTypeId=1 で収集可能である．

また，Duplicate で関係付けられた 2 つの文書では重複する内容であっても，Duplicate の関係を多段にたどっていくと，しだいに異なる内容になることがある．通常，FAQ 作成は FAQs に共通する内容に対して行われるため，本稿も複数の内容が FAQs 内に存在する場合は実験データから除外した．これは問合せクラスタのすべての問合せに共通するタグが存在するかどうかで判断した．たとえば，図 4 では，すべての文書に “r” というタグが付与されており，少なくともプログラミング言語 R という点で共通の内容であり，本稿が想定する実験データとしてふさわしい．これは Posts/PostsWithDeleted テーブル内の問合せ（PostType=1）の Tags 属性に設定されている．

以上より，次の条件にマッチする問合せクラスタを収集することとした．

- Duplicate 関係でつながっている問合せクラスタ
- クラスタサイズ（文書数）が 3 つ以上
- すべての問合せに共通するタグが 1 つ以上付与されていること

今回収集した問合せクラスタは 35,352 個，また，166,589 文書，2,032,504 単語で構成されていた．**図 5** に問合せクラスタのサイズごとの分布を示す．

6.1.2 教師データの作成

クラスタ内の文書の正解ランキングは，各問合せに付与された得票数を用いた．Stack Overflow では，問合せを読んだ利用者が有益と感じた場合は賛成票，そうでない場合は反対票を投じられるようになっており，その差分が各問合せの得票数となっている．たとえば，図 4 の文書は，それぞれ 238，0，−6 という得票数を獲得している．なお，得票数は Dump データの Posts/PostsWithDeleted テーブルでの問合せ（PostType=1）の Score 属性から収集可能である．

6.1.3 問合せ内容の前処理

問合せ内容を適切に抽出するために以下の前処理を行った．

処理 1：タイトル，本文，タグの抽出
処理 2：本文からコード箇所（<pre><code>と</pre></code>で囲まれた部分）を削除
処理 3：本文から Markdown をすべて削除
処理 4：タイトルと処理 2，3 で得られた本文の単語に対し，ストップワードを除去したうえでトークン列へ

© 2020 Information Processing Society of Japan

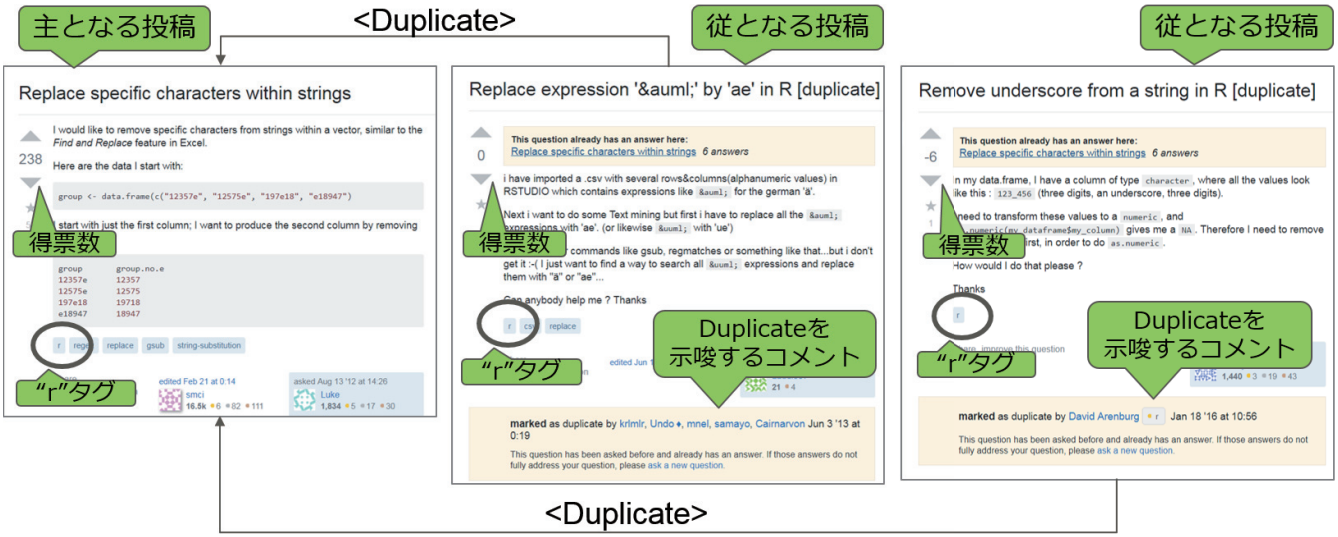

図 4 問合せクラスタの例
Fig. 4 An example of inquery cluster.

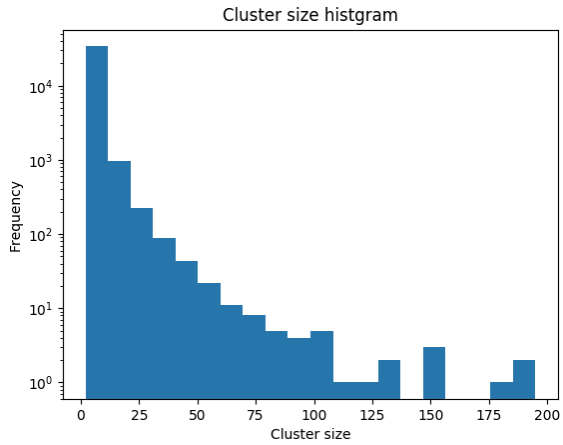

図 5 Stack Overflow から収集した問合せクラスタのサイズごとの分布
Fig. 5 Distribution of question-cluster size in Stack Overflow.

変換

問合せ内容は Dump データの Posts/PostsWithDeleted の PostTypeId=1 で収集でき，タイトル，本文，タグはそれぞれ Title，Body，Tags 属性に設定されている．Body 属性から抽出した本文は，プログラム断片や投稿文を構造的に描画するための Markdown [47] が埋め込まれているため，処理 2 や 3 を施した．

6.2 評価方法

本節では，クラスタ内文書ランキングの評価指標について説明する．クラスタ内文書ランキングは，4.2.4 項で説明した NDCG（Normalized Discounted Cumulative Gain）[35] に補正項 $\mathrm{DCG}_{\mathrm{worst}_j}$ を加えた A-NDCG（Adjusted-NDCG）を用いて評価することとした．

評価式は次のとおりである．

$$\mathrm{rate}_d = |\mathrm{cluster}_j| - \mathrm{rank}_d$$

$$\mathrm{DCG}_{\mathrm{cluster}_j} = \sum_{d \in \mathrm{cluster}_j} \frac{\mathrm{rate}_d}{\log_2(\mathrm{p_rank}_d + 1)}$$

$$\text{A-NDCG}_{\mathrm{cluster}_j} = \begin{cases} \text{nil} & (\text{if all } \mathrm{rate}_d \text{ values are the same}) \\ \frac{\mathrm{DCG}_{\mathrm{cluster}_j} - \mathrm{DCG}_{\mathrm{worst}_j}}{\mathrm{DCG}_{\mathrm{best}_j} - \mathrm{DCG}_{\mathrm{worst}_j}} & (\text{otherwise}) \end{cases}$$

ここで rank_d はクラスタ内での文書 d の正解順位を表し，文書 d が獲得した得票数で順序付けた．rate_d は，文書 d がクラスタ内で獲得した評価値で，クラスタ j のサイズから rank_d を引いたものとした．得票数ではなく正解順位を用いた理由は，得票数を直接評価値にしたときに，マイナスの取り扱いが困難であり，それを単純化するためである．たとえば，4 つの文書の評価値が $[\mathrm{rate}_{\mathrm{doc}_1}, \mathrm{rate}_{\mathrm{doc}_2}, \mathrm{rate}_{\mathrm{doc}_3}, \mathrm{rate}_{\mathrm{doc}_4}] = [-4, -3, -2, -1]$ のとき，$\mathrm{DCG}_{\mathrm{worst}}$ は -8.98 であるのに対し，$\mathrm{DCG}_{\mathrm{Best}}$ は -7.96 であり，$\mathrm{NDCG}_{\mathrm{worst}} = \frac{\mathrm{DCG}_{\mathrm{worst}}}{\mathrm{DCG}_{\mathrm{best}}}$ が $\mathrm{NDCG}_{\mathrm{best}} = \frac{\mathrm{DCG}_{\mathrm{best}}}{\mathrm{DCG}_{\mathrm{best}}}$ を超えてしまう．

$\mathrm{DCG}_{\mathrm{cluster}_j}$ はクラスタ j における DCG 値で，クラスタ内の各文書 d の保有スコアを予測順位 $\mathrm{p_rank}_d$ に応じた対数の減衰値の和を求めたものとなる．また，$\mathrm{DCG}_{\mathrm{best}_j}$ は $\mathrm{DCG}_{\mathrm{cluster}_j}$ の最大値，$\mathrm{DCG}_{\mathrm{worst}_j}$ は $\mathrm{DCG}_{\mathrm{cluster}_j}$ の最小値である．

$\mathrm{DCG}_{\mathrm{worst}_j}$ で補正する理由は，クラスタ内文書ランキングが通常の文書ランキングと異なり，ランキング結果に必ず関連する文書（評価値が 0 以上）が選択されるため，$\mathrm{DCG}_{\mathrm{worst}}$ が検索対象によって異なる値をとるためである．たとえば，2 つの文書クラスタの評価値がそれぞれ $[\mathrm{rate}_{\mathrm{doc}_1}, \mathrm{rate}_{\mathrm{doc}_2}, \mathrm{rate}_{\mathrm{doc}_3}, \mathrm{rate}_{\mathrm{doc}_4}] = [3, 2, 1, 0]$ と $[\mathrm{rate}_{\mathrm{doc}_1}, \mathrm{rate}_{\mathrm{doc}_2}, \mathrm{rate}_{\mathrm{doc}_3}] = [2, 1, 0]$ のとき，$\mathrm{NDCG}_{\mathrm{worst}}$ と $\mathrm{NDCG}_{\mathrm{best}}$ はそれぞれ 0.790〜1，0.788〜1 であり，クラスタ内の文書数によって NDCG 値の範囲が異なっている．

© 2020 Information Processing Society of Japan

これはクラスタ内文書ランキングに NDCG を適用すると正規化（Normalization）による性質が失われ，DCG 同様各ランキングを直接比較することができないことを示している．なお，上記の例を A-NDCG で評価すると補正の効果で 0〜1 の値をとるようになり，正規化が行われていることが分かる．

6.3 実験内容

本節では，実験内容について説明する．

6.3.1 実験の目的

実験では，本稿が提案する CTI2 および TI2 Filter がクラスタ内文書ランキングにおいて有効かどうかの評価を行う．その際，ICDF のパラメタ α および TI2 Filter でのパラメタ β や教師データを使った学習の有無が結果にどう影響するかも評価する．**表 1** は実験一覧になる．

ICDF のパラメタ α は，クラスタ内で低頻度の単語を除去する目的で設定されるパラメタであるため，今回の実験データに存在する最小クラスタを考慮して 1, 2 で実験する．

また，今回収集したデータで TF-IDF-ICDF を計測したところ，TF-IDF-ICDF$_{\alpha=1}$ の平均は 0.0197，分散は 0.0091，TF-IDF-ICDF$_{\alpha=2}$ の平均は 0.0073，分散は 0.0028 であった．**図 6** は実験データ内の TF-IDF-ICDF の分布になる．以上より TI2-Filter で用いる β は 0.01 刻みの 0 から 0.05 で実施する．

6.3.2 実験 1：学習なしメトリクスベース手法

実験 1 では，問合せのテキスト情報から抽出したメトリクスのみを使ったランキングを評価する．実験で用いたメトリクスは**表 2** のとおりである．

本実験は，Stack Overflow の得票数のような参考度を利用できない状況で，学習なしでランキングを行う場合を想定している．CTI2 やその他のさまざまなメトリクスを用いてクラスタ内文書ランキングの精度評価，およびベースラインに対する有効性を確認する．

ベースラインには，問合せクラスタからの非復元の無作為抽出によるランキングとした．これは FAQ 作成者が無作為に問合せを選択し，読解作業を進めることを想定した．また，有効性の確認方法は，ベースラインよりも平均精度が有意に高いかウェルチの t 検定 [53], [54], [55] で確認した．

6.3.3 実験 2：学習ありメトリクスベース手法

実験 2 は，問合せから抽出したメトリクスを入力に問合せの得票数を予測するモデルを学習し，その学習モデルから予測した得票数に基づいてランキングの精度を評価する．特徴量には表 2 のメトリクスを単一もしくは複数用いた．学習モデルには，線形，非線形モデルの代表的なモデルとして，それぞれ線形回帰モデル，多層パーセプトロン [56]（隠れ層の数 10，活性化関数は ReLu [57]）を採用した．

本実験では，実験 1 とは異なり参考度を活用できる状況を想定し，問合せから抽出したメトリクスと教師データを活用することで精度が改善するか確認する．また，全メトリクスに対して CTI2 を加えることで精度改善できるかも確認する．有効性の確認には実験 1 と同様にウェルチの t 検定を用いて検証する．

表 1 実験一覧
Table 1 Table of experiments.

実験	目的	学習	特徴量	パラメタ
1	CTI2 の有効性確認	なし	メトリクス	α
2	CTI2 の有効性確認	あり	メトリクス	α
3	TI2 Filter の有効性確認	あり	BOW	α，β

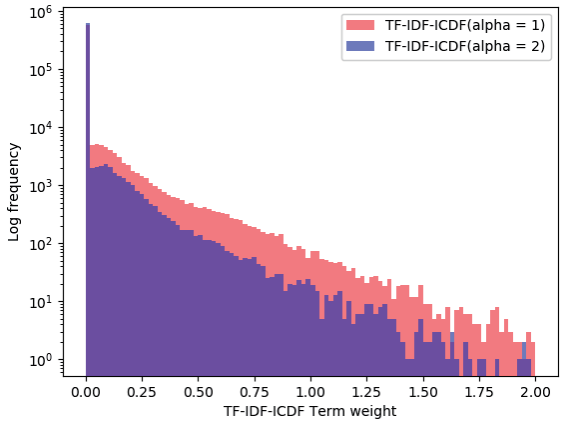

図 6 実験データ内の TF-IDF-ICDF の分布
Fig. 6 Term weight distribution of TF-IDF-ICDF in the experiment data.

6.3.4 実験 3：学習あり BOW ベース手法

実験 3 では，特徴量に問合せの BOW（Bag of Words）を用いて教師データからクラスタ内の文書のランキングを学習する手法を評価する．本手法では，実験 1，実験 2 で用いたメトリクスのような直感に頼った加工を行わず，問合せの BOW を直接入力することで情報損失を防ぎつつ，精度向上に寄与する特徴を選択・作成できる可能性がある．学習モデルには**図 7** に示す多層パーセプトロンを用いる．入力層に単語数，隠れ層に 50 個および 1 個のニューロンから構成されるネットワークを用意し，隠れ層の活性化関数は ReLu を採用した．それを Stack Overflow のタイトル，本文，タグの BOW を個別に入力し，3 つの出力結果を引数にした sigmoid 関数で文書の重要度を出力した．文書の重要度は，文書に付与された得票数が 0 以下の場合 0，1 から 6 のとき 0.5，7 以上のとき 1 として学習を行った．

この推論（フィードフォワード）時の計算量は，(1) 入力層から出力層への行列計算と，(2) 活性化関数および sigmoid 関数の計算の和となる．したがって，入力層の数（単語数）を n，隠れ層の数を 50，出力層の数を 1 および項目（タイトル，本文，タグ）の数を 3，活性化関数 ReLu および sigmoid 関数の計算量を 1 とすると，計算量

© 2020 Information Processing Society of Japan

表 2 ベースラインおよび学習なし手法で用いるメトリクスの一覧
Table 2 Baseline method and unsupervised learning methods.

カテゴリ	メトリクス	説明
Baseline Method	-	問合せクラスタから問合せを非復元で無作為抽出し，抽出した順番でランキング
Textual Metrics	Body Length	ソースコードや HTML のタグを含む質問の長さ
	Emails Count	質問中に含まれる Email の数
	Lowercase Percentage	質問中の小文字の数の割合
	Uppercase Percentage	質問文中の大文字の割合
	Spaces Count	質問中のスペースの数
	Tags Count	質問に付与されたタグの数
	Text Speak Count	メールや Twitter などで使われる省略文字の数
	Title Body Similarity	タイトルと本文の類似度（本稿では Cosine Similarity とした）
	Title Length	タイトルの長さ
	Capital Title	タイトルが大文字で始まっているかどうか（0 or 1）
	URLs Count	質問中の URL の数
	Code Length	質問中のコードの長さ
	LOC Percentage	全行数中のコード行数の割合
	Words Count	コードやタグを除く単語の数
	Sentences Count	コードやタグを除く質問の行数
	Metric Entropy	(shannon entropy)/(body length)
	Cumulative TF-IDF	質問中のすべての単語の TF-IDF の総和
Readability Metrics	Automated Reading Index [48]	$4.71 \cdot \frac{\text{characters}}{\text{words}} + 0.5 \cdot \frac{\text{words}}{\text{sentences}} - 21.43$
	Coleman Liau Index [49]	$0.588 \cdot \mathrm{L} - 0.296 \cdot \mathrm{S} - 15.8$ L：100 単語ごとの平均文字数，S：100 単語ごとの平均文章数
	Flesch Kincaid Grade Level [50]	$0.39 \cdot \frac{\text{total words}}{\text{total sentences}} + 11.8 \cdot \frac{\text{total syllables}}{\text{total words}} - 11.9$
	Flesch Reading Ease Score [50]	$206.835 - 1.015 \cdot \frac{\text{total words}}{\text{total sentences}} - 84.6 \cdot \frac{\text{total syllables}}{\text{total words}}$
	Gunning Fog Index [51]	$0.4 \cdot \text{words_sentences} + 100 \cdot \frac{\text{complex words}}{\text{words}}$
	SMOG Grade [52]	$1.0430 \cdot \sqrt{\text{polysyllables} \cdot \frac{30}{\text{sentences}}} + 3.1291$
Our Methods	$\text{CTI2}_{\alpha=1}$(Cumulative TF-IDF-$\text{ICDF}_{\alpha=1}$)	質問中のすべての単語の TF-IDF-$\text{ICDF}_{\alpha=1}$ の総和
	$\text{CTI2}_{\alpha=1}$(Cumulative TF-IDF-$\text{ICDF}_{\alpha=2}$)	質問中のすべての単語の TF-IDF-$\text{ICDF}_{\alpha=2}$ の総和

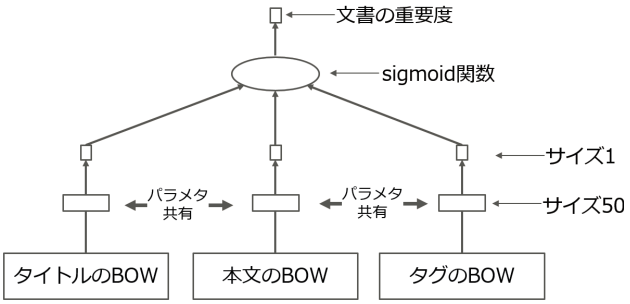

図 7 実験 3 で用いた学習モデル
Fig. 7 Learning model used in experiment 3.

は $\underbrace{(n*50+50*1)*3}_{(1)} + \underbrace{(50+1)*3+1}_{(2)}$ となり，$\mathcal{O}(n)$ と見なせる．

本手法は，入力特徴量に問合せの BOW を用いることで高精度化が期待できる．一方で入力次元数が大きいため計算量やメモリ量が大きくなる懸念がある．これを改善するために入力特徴量のうち，重要な特徴量を残しつつ精度維持もしくは改善できる特徴量選択が必要となる．本実験では，本手法による精度改善および TI2 Filiter による特徴量選択が精度にどのような影響を与えるか確認する．

6.4 実装

実装言語は Python3.5，機械学習ライブラリは Scikit-learn [58]，Keras [59]，前処理には NLTK [60] を用いた．また，実行環境として，AWS の p2.xlarge [61] を用いた．

7. 実験結果と考察

本節では，実験結果と考察を説明する．

7.1 実験 1 の結果と考察

実験 1 の結果は**表 3** のとおりである．

学習なしメトリクスベース手法での最高精度は，Exp1-6（Tags Count）と Exp1-25（$\text{CTI2}_{\alpha=1}$）で確認され，A-NDCG 0.563 あった．Exp1-25（$\text{CTI2}_{\alpha=1}$）と Exp1-26（$\text{CTI2}_{\alpha=2}$）からは Exp1-1（Baseline method）からの有意差も確認できた．したがって，CTI2 を用いることでランキングの高精度化を実現できることが確認できた．

© 2020 Information Processing Society of Japan

表 3 実験 1 の結果
Table 3 Results of experiment 1.

実験番号	カテゴリ	メトリクス	A-NDCG	検定の比較対象	p-value
Exp1-1	Baseline Method	-	0.505	-	-
Exp1-2	Textual Metrics	Body Length	0.489	Exp1-1	4.70e-4**
Exp1-3		Emails Count	0.506	Exp1-1	0.506
Exp1-4		Lowercase Percentage	0.504	Exp1-1	0.648
Exp1-5		Spaces Count	0.469	Exp1-1	0.073
Exp1-6		Tags Count	0.563	Exp1-1	1.65e-128**
Exp1-7		Text Speak Count	0.510	Exp1-1	0.047*
Exp1-8		Title Body Similarity	0.518	Exp1-1	4.35e-7**
Exp1-9		Title Length	0.497	Exp1-1	0.055
Exp1-10		Capital Title	0.529	Exp1-1	1.16e-30**
Exp1-11		Uppercase Percentage	0.510	Exp1-1	0.041*
Exp1-12		URLs Count	0.504	Exp1-1	0.393
Exp1-13		Code Length	0.463	Exp1-1	4.97e-27**
Exp1-14		LOC Percentage	0.461	Exp1-1	9.08e-74**
Exp1-15		Words Count	0.515	Exp1-1	0.444
Exp1-16		Sentences count	0.510	Exp1-1	0.471
Exp1-17		Metric Entropy	0.515	Exp1-1	1.66e-5**
Exp1-18		Cumulative TF-IDF	0.485	Exp1-1	2.74e-14**
Exp1-19	Readability Metrics	Automated Reading Index	0.508	Exp1-1	0.363
Exp1-20		Coleman Liau Index	0.502	Exp1-1	0.444
Exp1-21		Flesch Kincaid Grade Level	0.510	Exp1-1	0.092
Exp1-22		Flesch Reading Ease Score	0.507	Exp1-1	0.483
Exp1-23		Gunning Fog Index	0.515	Exp1-1	1.63e-5**
Exp1-24		SMOG Grade	0.513	Exp1-1	0.312
Exp1-25	Our Method	$CTI2_{\alpha=1}$	0.563	Exp1-1	3.03e-106**
Exp1-26		$CTI2_{\alpha=2}$	0.551	Exp1-1	8.79e-71**
				Exp1-25	4.92e-05**

また, Exp1-18（Cumulative TF-IDF）と比較して, Exp1-25（$CTI2_{\alpha=1}$）と Exp1-26（$CTI2_{\alpha=2}$）ともに大きな精度改善が見られ，TF-IDF へ ICDF を組み込むことで精度改善できることを確認できた．

Exp1-25（$CTI2_{\alpha=1}$）と Exp1-26（$CTI2_{\alpha=2}$）の比較からは，α の決め方はクラスタサイズを考慮する必要があることが分かった．Exp1-26 ではすべてのクラスタに共通して頻度 2 以下の単語を除去したが，クラスタサイズ 3 のような小さい問合せクラスタに対しては高頻度な単語を除去してしまったことになる．これが Exp1-25 に対して精度悪化を招いた可能性がある．今後，クラスタサイズを考慮した α の設定方法を検討したい．

Tags Count（Exp1-6）が高精度である理由は，文書の評価値とタグ数の間に文書の非閲覧回数を介した疑似的な相関関係があるためと考える．高く評価される文書は何度も閲覧され，自ずとレビューを受ける機会が増える．レビュー回数が多いとタグが見直される機会も増え，タグの数が増える傾向にあると考えられるためである．なお，タグは Stack Overflow の投稿者，サイト管理者，保守者が付与・整備したことにより作成された，いわば人手によって作成された情報である．それに対し，CTI2（Exp1-25, 26）は人手によらずテキストから抽出可能なメトリクスであり，適用範囲が広いといえる．

以上により学習なしメトリクスベースのクラスタ内文書ランキングにおいて，CTI2 は有効なメトリクスであることが確認できた．

7.2 実験 2 の結果と考察

実験 2 の結果は**表 4** のとおりである．単一メトリクスを用いた実験は，実験 1 で高精度だったメトリクス（Tags Count，$CTI2_{\alpha=1}$，$CTI2_{\alpha=2}$）を含め，学習による精度改善は見られなかった．また，Exp2-4〜6 における複数のメトリクスを入力にした線形回帰モデルにおいても，CTI2 による有意な精度改善は確認できなかった．一方，Exp2-10〜12 における複数のメトリクスを入力にした多層パーセプトロンでは，CTI2 によって有意に精度改善できることを確認した．以上より，非線形モデルを用いて複数のメトリクスを活用する場合で CTI2 の有効性が確認できた．

7.3 実験 3 の結果と考察

実験 3 では，多層パーセプトロンベースで BOW を入力に精度評価を行った．入力に用いる BOW は以下の 3 ケースで評価した．

- すべての単語を用いる場合
- ランダムに単語を選択した場合
- TI2-Filter で α が 0 と 1，β が 0.01 刻みの 0 から 0.05

© 2020 Information Processing Society of Japan

表 4 実験 2 の結果
Table 4 Results of experiment 2.

実験番号	学習モデル	特徴量	A-NDCG	検定の比較対象	p-value
Exp2-1	線形回帰モデル	Tags Count	0.563	Exp1-6	1.0
Exp2-2		$CTI2_{\alpha=1}$	0.563	Exp1-25	1.0
Exp2-3		$CTI2_{\alpha=2}$	0.551	Exp1-26	1.0
Exp2-4		Textual_Metrics + Readability_Metrics	0.585	-	-
Exp2-5		Textual_Metrics + Readability_Metrics + $CTI2_{\alpha=1}$	0.587	Exp2-4	0.929
Exp2-6		Textual_Metrics + Readability_Metrics + $CTI2_{\alpha=2}$	0.586	Exp2-4	0.786
Exp2-7	多層パーセプトロン (隠れ層のサイズ 10)	Tags Count	0.563	Exp1-6	1.0
Exp2-8		$CTI2_{\alpha=1}$	0.563	Exp1-25	1.0
Exp2-9		$CTI2_{\alpha=2}$	0.551	Exp1-26	1.0
Exp2-10		Textual_Metrics + Readability_Metrics	0.585	-	-
Exp2-11		Textual_Metrics + Readability_Metrics +$CTI2_{\alpha=1}$	0.598	Exp2-10	1.88e-05**
Exp2-12		Textual_Metrics + Readability_Metrics +$CTI2_{\alpha=2}$	0.599	Exp2-10	1.00e-07**
				Exp2-11	0.296

表 5 実験 3 の結果
Table 5 Results of experiment 3.

実験番号	単語の選び方	入力層	A-NDCG
Exp3-1	全単語	2,032,504	0.693
Exp3-2	ランダム	60,549	0.554
Exp3-3	TI2-$Filter_{\alpha=1,\beta=0}$	60,549	0.702
Exp3-4	TI2-$Filter_{\alpha=1,\beta=0.01}$	45,735	0.701
Exp3-5	TI2-$Filter_{\alpha=1,\beta=0.02}$	35,101	0.702
Exp3-6	TI2-$Filter_{\alpha=1,\beta=0.03}$	27,696	0.702
Exp3-7	TI2-$Filter_{\alpha=1,\beta=0.04}$	22,228	0.703
Exp3-8	TI2-$Filter_{\alpha=1,\beta=0.05}$	18,303	0.691
Exp3-9	TI2-$Filter_{\alpha=2,\beta=0}$	26,405	0.700
Exp3-10	TI2-$Filter_{\alpha=2,\beta=0.01}$	19,373	0.700
Exp3-11	TI2-$Filter_{\alpha=2,\beta=0.02}$	14,793	0.701
Exp3-12	TI2-$Filter_{\alpha=2,\beta=0.03}$	11,804	0.699
Exp3-13	TI2-$Filter_{\alpha=2,\beta=0.04}$	9,596	0.697
Exp3-14	TI2-$Filter_{\alpha=2,\beta=0.05}$	7,873	0.678

の値で単語を選択した場合

実験結果は**表 5** のとおりである．

実験の結果，TI2-$Filter_{\alpha=1,\beta=0.04}$（Exp3-7）において最高精度 A-NDCG 0.703 を達成した．Exp3-1 と Exp3-13 の比較からは，TI2-Filter によって約 1/200 に計算量を削減しつつ精度維持を実現できることが分かった．なお，6.3.4 項で述べたとおり，本実験での計算量は入力層のサイズに比例する．また，TI2-Filter を使って単語選択を行うことで Exp3-8 と Exp3-14 を除いた実験で Exp3-1 を上回る精度が確認でき，TI2-Filter が高精度化に寄与することが分かった．Exp3-8 と Exp3-14 は，閾値 β を最も大きくした結果，精度向上に寄与する単語まで除外してしまった可能性が考えられる．

一方，Exp3-2 と Exp3-3 は，計算量は同等であるが，Exp3-2 が Exp3-3 に対し大きく精度悪化している．この結果から，TI2-Filter が高精度化に寄与する単語を選択できていることが分かった．

また，$\alpha = 1$ を設定した実験（Exp3-3〜8）と $\alpha = 2$ を設定した実験（Exp3-9〜14）を比較すると，おおむね $\alpha = 1$ のほうが高精度であり，実験 1 での議論と同様に，クラスタ内の単語除去によって副作用が起きている可能性がある．

以上より，TI2 Filiter による適切な特徴量選択によって，計算量を削減しつつ，精度改善が行えることが分かった．

8. 今後の課題

本節では今後の課題について説明する．

8.1 本手法の精度改善

本稿では Stack Overflow を使った提案手法の有効性の確認を行った．今後，他のデータセットでも有効性を確認したい．たとえば，システムやネットワークに関する Q&A サイト [62] などは，Stack Overflow と同様のデータスキーマで管理されており，同様の実験手順で即座に評価できる．

また，さらなる高精度化に向けて，さまざまな特徴量や学習モデルの活用を検討したい．たとえば，図 7 の sigmoid 関数の引数に各項目の CTI2 を組み込むなど，CTI2 と TI2 Filter 後の特徴量を同時に活用する方法も考えられる．また，Yao ら [28] は Stack Overflow で質問と回答の得票数に相関があることや，受理された回答（Accepted Answer）の有無が質問の得票数に影響することを示唆しており，特徴量設計の参考にしたい．さらに，今回の実験で実施できなかった α や β の組合せでの実験も検討したい．

8.2 実用化に向けた改善

本節では，クラスタ内文書ランキングを FAQ 作成に実用していく際の問題を説明する．

まず，現状のクラスタ内文書ランキングでは，単にランキング結果のみを提示するので，上位からどこまで読み進めれば十分か分からない点が問題といえる．利用者がすべ

© 2020 Information Processing Society of Japan

ての文書を読んでしまうと，本稿の目的である読解量の削減を実現できないためである．これを避けるために，たとえば，読み進めていく過程で獲得した情報量に基づいて読解の停止箇所を示す方法が考えられる．

また，問合せが頻出する前にFAQ作成が完了していないことがありうることも問題といえる．コストをかけてFAQを作成しても活用できる機会がなければ意味がないためである．これは問合せが頻発する可能性を事前に予測・判別することで回避できる可能性がある．たとえば，OSのセキュリティアップデート直後は類似する問合せが頻発しやすく，問合せ内容に基づく判別を行うことでFAQ作成を事前に済ませておくことが可能になる．

9. 妥当性への脅威

本節では，評価実験に含まれる妥当性への脅威について説明する．

本稿の実験では，Stack Overflowから抽出されたクラスタや教師データを用いた．その際，本稿が定めたクラスタの選別条件や教師データはStack Overflowの利用者や管理者によって付与された情報に基づいており，誤っている可能性がある．また，本稿ではStack Overflowを用いた実験を実施したが，これと異なる実験データを用いた場合，必ずしも同様の結果にならない可能性がある．

10. まとめ

本稿では，限られたリソースでクラウド文書からソフトウェア文書へ情報を反映すべく，FAQsから読むべき優先度の高い文書をランキングする方式を検討・評価した．具体的には，文書クラスタが所与であることを活かして，文書クラスタごとに適切な単語の重み付けを行うICDFを開発した．またICDFを応用したCTI2およびTI2 Filterを開発し，有効性を確認した．とりわけTI2 Filterを用いた学習あり手法においては，A-NDCG 0.703を達成した．

参考文献

[1] Bauer, B.J. and Parnas, D.L.: Applying Mathematical Software Documentation: An Experience Report, *COMPASS'95 Proc. 10th Annual Conference on Computer Assurance Systems Integrity, Software Safety and Process Security'*, pp.273–284, IEEE (1995).

[2] Briand, L.C.: Software documentation: How much is enough?, *Proc. 7th European Conference on Software Maintenance and Reengineering*, pp.13–15, IEEE (2003).

[3] Jiau, H.C. and Yang, F.-P.: Facing up to the inequality of crowdsourced API documentation, *ACM SIGSOFT Software Engineering Notes*, Vol.37, No.1, pp.1–9 (2012).

[4] Parnin, C., Treude, C., Grammel, L. and Storey, M.-A.: Crowd documentation: Exploring the coverage and the dynamics of API discussions on Stack Overflow, Georgia Institute of Technology, Technical Report (2012).

[5] Howe, J.: The Rise of Crowdsourcing, Mozilla Foundation (online), available from ⟨https://www.wired.com/2006/06/crowds/⟩.

[6] Foundation, W.: Wikipeida, Wikimedia Foundation (online), available from ⟨https://www.wikipedia.org/⟩.

[7] Stack Exchange, I.: Stack Exchange Data Dump, Stack Exchange, Inc. (online), available from ⟨https://archive.org/details/stackexchange⟩.

[8] 国立情報学研究所：トップエスイー，国立情報学研究所（オンライン），入手先 ⟨https://www.topse.jp/ja/⟩.

[9] Community, E.: Eclipse Community Forums, Eclipse Foundation (online), available from ⟨https://www.eclipse.org/forums/⟩.

[10] Bugzilla: Bugzilla, Mozilla Foundation (online), available from ⟨https://bugzilla.mozilla.org/home⟩.

[11] Stack Exchange, I.: Stack Overflow, Stack Exchange, Inc. (online), available from ⟨https://stackoverflow.com/⟩.

[12] Sun, C., Lo, D., Khoo, S.-C. and Jiang, J.: Towards more accurate retrieval of duplicate bug reports, *Proc. 2011 26th IEEE/ACM International Conference on Automated Software Engineering*, pp.253–262, IEEE Computer Society (2011).

[13] Sun, C., Lo, D., Wang, X., Jiang, J. and Khoo, S.-C.: A discriminative model approach for accurate duplicate bug report retrieval, *Proc. 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pp.45–54, ACM (2010).

[14] Zhang, Y., Lo, D., Xia, X. and Sun, J.-L.: Multi-factor duplicate question detection in stack overflow, *Journal of Computer Science and Technology*, Vol.30, No.5, pp.981–997 (2015).

[15] Ahasanuzzaman, M., Asaduzzaman, M., Roy, C.K. and Schneider, K.A.: Mining duplicate questions in stack overflow, *Proc. 13th International Conference on Mining Software Repositories*, pp.402–412, ACM (2016).

[16] Mizobuchi, Y. and Takayama, K.: Two improvements to detect duplicates in Stack Overflow, *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering* (*SANER*), pp.563–564, IEEE (2017).

[17] de Souza, L.B., Campos, E.C. and Maia, M.D.A.: Ranking crowd knowledge to assist software development, *Proc. 22nd International Conference on Program Comprehension*, pp.72–82, ACM (2014).

[18] Xu, B., Xing, Z., Xia, X. and Lo, D.: AnswerBot: Automated generation of answer summary to developers' technical questions, *2017 32nd IEEE/ACM International Conference on Automated Software Engineering* (*ASE*), pp.706–716, IEEE (2017).

[19] Gao, Q., Zhang, H., Wang, J., Xiong, Y., Zhang, L. and Mei, H.: Fixing recurring crash bugs via analyzing Q&A sites (T), *2015 30th IEEE/ACM International Conference on Automated Software Engineering* (*ASE*), pp.307–318, IEEE (2015).

[20] Hu, W.-C., Yu, D.-F. and Jiau, H.C.: A faq finding process in open source project forums, *2010 5th International Conference on Software Engineering Advances*, pp.259–264, IEEE (2010).

[21] Henß, S., Monperrus, M. and Mezini, M.: Semiautomatically extracting FAQs to improve accessibility of software development knowledge, *Proc. 34th International Conference on Software Engineering*, pp.793–803, IEEE Press (2012).

[22] Iwai, K., Iida, K., Akiyoshi, M. and Komoda, N.: A help

© 2020 Information Processing Society of Japan

desk support system with filtering and reusing e-mails, *2010 8th IEEE International Conference on Industrial Informatics*, pp.321–325, IEEE (2010).
[23] Bozdogan, C. and Zincir-Heywood, N.: Data mining for supporting it management, *2012 IEEE Network Operations and Management Symposium*, pp.1378–1385, IEEE (2012).
[24] Chen, C. and Zhang, K.: Who asked what: Integrating crowdsourced faqs into api documentation, *Companion Proc. 36th International Conference on Software Engineering*, pp.456–459, ACM (2014).
[25] Bihani, A., Ullman, J.D. and Paepcke, A.: FAQtor: Automatic FAQ generation using online forums, Technical Report, Stanford InfoLab. (2018).
[26] Ponzanelli, L., Mocci, A., Bacchelli, A., Lanza, M. and Fullerton, D.: Improving low quality stack overflow post detection, *2014 IEEE International Conference on Software Maintenance and Evolution*, pp.541–544, IEEE (2014).
[27] Xia, X., Lo, D., Correa, D., Sureka, A. and Shihab, E.: It takes two to tango: Deleted stack overflow question prediction with text and meta features, *2016 IEEE 40th Annual Computer Software and Applications Conference* (*COMPSAC*), Vol.1, pp.73–82, IEEE (2016).
[28] Yao, Y., Tong, H., Xie, T., Akoglu, L., Xu, F. and Lu, J.: Detecting high-quality posts in community question answering sites, *Information Sciences*, Vol.302, pp.70–82 (2015).
[29] Chaparro, O., Lu, J., Zampetti, F., Moreno, L., Di Penta, M., Marcus, A., Bavota, G. and Ng, V.: Detecting missing information in bug descriptions, *Proc. 2017 11th Joint Meeting on Foundations of Software Engineering*, pp.396–407, ACM (2017).
[30] Di Sorbo, A., Panichella, S., Visaggio, C.A., Di Penta, M., Canfora, G. and Gall, H.C.: Development emails content analyzer: Intention mining in developer discussions (T), *2015 30th IEEE/ACM International Conference on Automated Software Engineering* (*ASE*), pp.12–23, IEEE (2015).
[31] Sparck Jones, K.: A statistical interpretation of term specificity and its application in retrieval, *Journal of documentation*, Vol.28, No.1, pp.11–21 (1972).
[32] Jones, K.S.: Index term weighting, *Information storage and retrieval*, Vol.9, No.11, pp.619–633 (1973).
[33] Beel, J., Gipp, B., Langer, S. and Breitinger, C.: paper recommender systems: A literature survey, *International Journal on Digital Libraries*, Vol.17, No.4, pp.305–338 (2016).
[34] Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E.D., Gutierrez, J.B. and Kochut, K.: A brief survey of text mining: Classification, clustering and extraction techniques, arXiv preprint arXiv:1707.02919 (2017).
[35] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N. and Hullender, G.N.: Learning to rank using gradient descent, *Proc. 22nd International Conference on Machine learning* (*ICML-05*), pp.89–96 (2005).
[36] Baeza-Yates, R., Ribeiro-Neto, B., et al.: *Modern information retrieval*, Vol.463, ACM press New York (1999).
[37] Voorhees, E.M. et al.: The TREC-8 question answering track report, *Trec*, Vol.99, pp.77–82, Citeseer (1999).
[38] Fuhr, N.: Some common mistakes in IR evaluation, and how they can be avoided, *ACM SIGIR Forum*, Vol.51, No.3, pp.32–41, ACM (2018).
[39] PubMed, US National Library of Medicine National Institutes of Health (online), available from ⟨https://www.ncbi.nlm.nih.gov/pubmed/⟩.
[40] Islamaj, D.R., Murray, G.C., Névéol, A. and Lu, Z.: Understanding PubMed® user search behavior through log analysis, *Database*, Vol.2009 (2009).
[41] Granka, L.A., Joachims, T. and Gay, G.: Eye-tracking analysis of user behavior in WWW search, *Proc. 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.478–479, ACM (2004).
[42] Järvelin, K. and Kekäläinen, J.: IR evaluation methods for retrieving highly relevant documents, *Proc. 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.41–48, ACM (2000).
[43] Guyon, I. and Elisseeff, A.: An introduction to variable and feature selection, *Journal of Machine Learning Research*, Vol.3, No.Mar, pp.1157–1182 (2003).
[44] wiki, C.: Database schema documentation for the public data dump and SEDE, Stack Exchange, Inc. (online), available from ⟨https://meta.stackexchange.com/questions/2677/database-schema-documentation-for-the-public-data-dump-and-sede/2678#2678⟩.
[45] Stack Exchange, I.: Why are some questions marked as duplicate?, Stack Exchange, Inc. (online), available from ⟨https://stackoverflow.com/help/duplicates⟩.
[46] Ihaka, R. and Gentleman, R.: R: A Language for Data Analysis and Graphics, *Journal of Computational and Graphical Statistics*, Vol.5, No.3, pp.299–314 (1996).
[47] Stack Exchange, I.: Markdown help, Stack Exchange, Inc. (online), available from ⟨https://stackoverflow.com/editing-help⟩.
[48] Senter, R. and Smith, E.A.: Automated readability index, Technical Report, CINCINNATI UNIV OH (1967).
[49] Coleman, M. and Liau, T.L.: A computer readability formula designed for machine scoring, *Journal of Applied Psychology*, Vol.60, No.2, p.283 (1975).
[50] Kincaid, J.P., Fishburne Jr, R.P., Rogers, R.L. and Chissom, B.S.: Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel (1975).
[51] Gunning, R.: The technique of clear writing (1952).
[52] Mc Laughlin, G.H.: SMOG grading-a new readability formula, *Journal of reading*, Vol.12, No.8, pp.639–646 (1969).
[53] Welch, B.L.: The generalization ofstudent's' problem when several different population variances are involved, *Biometrika*, Vol.34, No.1/2, pp.28–35 (1947).
[54] Ruxton, G.D.: The unequal variance t-test is an underused alternative to Student's t-test and the Mann–Whitney U test, *Behavioral Ecology*, Vol.17, No.4, pp.688–690 (2006).
[55] Derrick, B., Toher, D. and White, P.: Why Welch's test is Type I error robust, *The Quantitative Methods in Psychology*, Vol.12, No.1, pp.30–38 (2016).
[56] Bishop, C.M.: *Pattern recognition and machine learning*, Springer (2006).
[57] LeCun, Y., Bengio, Y. and Hinton, G.: Deep learning, *nature*, Vol.521, No.7553, p.436 (2015).
[58] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research*, Vol.12, No.Oct, pp.2825–2830 (2011).
[59] Chollet, F.: Keras (2015).

© 2020 Information Processing Society of Japan

[60] Bird, S., Klein, E. and Loper, E.: *Natural language processing with Python: Analyzing text with the natural language toolkit*, O'Reilly Media, Inc. (2009).
[61] Amazon：Amazon EC2 p2 インスタンス，Amazon（オンライン），入手先 ⟨https://aws.amazon.com/jp/ec2/instance-types/p2/⟩.
[62] Stack Exchange, I.: Server Fault, Stack Exchange, Inc. (online), available from ⟨https://serverfault.com/⟩.

溝渕 裕司 （正会員）

1980 年生．2003 年東京工業大学理学部情報科学科卒業．2005 年同大学大学院数理計算科学科修了．同年株式会社富士通研究所入社．ソフトウェア工学，テキストマイニングの研究に従事．人工知能学会会員．

吉田 和樹 （正会員）

1964 生．1987 年東京工業大学工学部経営工学科卒業．1989 年同大学大学院総合理工学研究科システム科学専攻修士課程修了．同年株式会社東芝入社．ソフトウェア生産技術の研究・開発・普及活動に従事．1995～1997 年イリノイ大学客員研究員，2005 年東京工業大学社会理工学研究科経営工学専攻社会人博士課程修了，博士（工学）．2007～2008 年東京大学非常勤講師，2018 年より国立情報学研究所 GRACE センター特任研究員．経営情報学会会員．

© 2020 Information Processing Society of Japan