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)]　吉田 和樹[2,b)]

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（ソフトウェア文書と呼ぶ）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，クラウド文書に記載されたナレッジを活用することで解決することがあり，広く利用されている．クラウド文書のうち頻出する問合せ (FAQs) は，マニュアル (FAQ) 化およびソフトウェア文書への反映によってクラウド文書への問合せが減らせ，作業の効率化を期待できる．しかしながら，FAQ 化作業は事前に問合せ文書の読解が必要であり，リソースが限られる中，全ての文書を読解することは不可能である．本稿では，過去の膨大な問合せ文書から効率的かつ効果的に内容把握するために，優先的に読むべき文書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，これを応用したクラスタ内文書ランキングに適用することで A-NDCG 0.701 を達成することができた．

1. はじめに

ソフトウェアドキュメンテーションとは，ソフトウェアに付随する文書を作成することであり，各ステークホルダーが必要な情報を文書化することである．ソフトウェア文書には，例えば，要件定義書，設計書，テスト仕様書，API ドキュメント，利用手引き等がある．

一般にソフトウェア文書は，ソフトウェアの管理母体が正式に公開しているもので，ソフトウェア利用者が初めに参照する文書である．従って，可能な限り正確かつ網羅的に整備されていることが望まれる [1], [2]．しかしながら，ソフトウェア文書は常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させていく必要がある．

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，クラウド文書と呼ばれるものが利用されている [3], [4], [5]．例えば，E-mail（特にメーリングリスト），OSS コミュニティフォーラムでのスレッド，Q&A サイトの質疑応答，また Wikipedia[6] などのクラウドソーシングによる文書がある．クラウド文書はソフトウェア管理母体とは異なる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，冗長性等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，有用なナレッジ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の情報反映が行われることが望ましい．なかでも多数の類似するクラウド文書は，多くの利用者がソフトウェア文書に同じ情報を問合せて失敗した結果であり，早急な情報反映によって以後の作業効率化が可能である．

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群 (FAQs; Frequently Asked Questions) をマニュアル (FAQ; Frequently Asked Question) 化する作業の効率化を検討する．FAQ 化作業は，概して，(1)FAQs の選出，(2) 内容把握，(3) 内容の整理・文書化から構成され，膨大な過去の履歴からの作成には多大なコストが必要になる．例えば，2018 年 9 月時点の Stack Overflow Dump Data[7] から抽出した文書数が 3 つ以上の問合せクラスタは 35352 個あり，そのうち最も大きいクラスタは 195 文書から構成されていた．とりわけ内容把握では，作業者の読解量に限りがある中，いかに重要な文書を選択し読むべき範囲を絞り込んでいくかが効率化のポイントとなる．

本稿では，限られたリソースで FAQ 化作業を実現するべく，FAQs から読むべき優先度の高い文書をランキング

[1] 株式会社富士通研究所
[2] 国立情報学研究所
[a)] mizobuchi.yuji@fujitsu.com
[b)] k-yoshida@nii.ac.jp

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) ランキング対象が類似する文書群である点，が特徴と言える．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，一般の文書クラスタリングより高精度化が難しいと言える．なお，当方が調査した限りでは，同様の問題設定を過去に報告された実績はなく，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，問題の特性上，事前に文書クラスタが所与であることを前提とできるため，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) を応用したもので，クラスタ内の単語の出現頻度を考慮したものである．さらに ICDF を活用し，文書重要度の算出手法 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 0.701 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. 関連研究・準備

この節では関連研究について説明する．

2.1 クラウド文書の利活用

従来から Eclipse Community Forums[9] や Bugzilla[10] をはじめとする OSS コミュニティフォーラムでは，バグや使用方法について議論が交わされてきた．また，Stack Overflow[11] などのナレッジコミュニティでは，バグレポートや API の使用法など様々な内容で質問回答が繰り返されてきた．

OSS コミュニティフォーラムでのナレッジ利活用に関する研究には，過去に投稿されたバグレポートを高精度に検索する方法 [12], [13] や，バグレポートにバグの現象や再現方法が記載されているか分析する方法 [14] などがある．

また，ナレッジコミュニティでのナレッジ利活用に関する研究には，Stack Overflow での過去の投稿を高精度に検索する方法 [15], [16], [17] や，質問タイプの判別 [18]，回答候補の検索および要約 [19]，過去のバグレポートから修正コードの生成 [20]，Stack Overflow 内の FAQ と API ドキュメントの関連付 [21] などがある．

FAQ 化の作業支援に関しては，OSS コミュニティフォーラムでの FAQ 化作業支援 [22], [23]，ヘルプデスクでの FAQ 化作業および活用支援 [24]，IT システムでのトラブル対処方法の FAQ 化作業支援 [25]，オンライン学習サービスでの FAQ 化作業支援 [26] などがある．

その他，E-mail の内容分析 [27], バグレポートの内容判別 [14] などがある．

何れの先行研究も FAQ 化作業の際の読解支援を対象としていない．

2.2 Stack Overflow の投稿内容の品質評価

2.1 節では，Stack Overflow の投稿内容を利活用する関連研究を説明したが，その他にも Stack Overflow での投稿内容の品質管理に関する研究がある．

Ponzanelli ら [28] は，Stack Overflow に投稿される低品質な質問・回答を自動で特定するためにテキストメトリクスや読解力メトリクスを活用して，得票数 (Stack Overflow では Vote と呼ばれる) に基づく品質レベルを予測するモデルを提案した．Xin ら [29] も同様に投稿内容の品質測定を行い，品質に起因して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 [30] は，質問と回答の獲得投票数に相関があることに注目し，これを活用した予測方法を提案した．何れの手法も対象文書群に特段の選別を行っていない．

一方，クラスタ内文書ランキングの対象文書群は類似する文書間の差異として使える特徴量が少ない．図 1 はクラスタ内文書間のコサイン類似度とランダムにサンプリングした文書間のコサイン類似度の分布を示している．クラスタ内の文書間のコサイン類似度はランダムにサンプルした場合に比べて高いことが分かる．

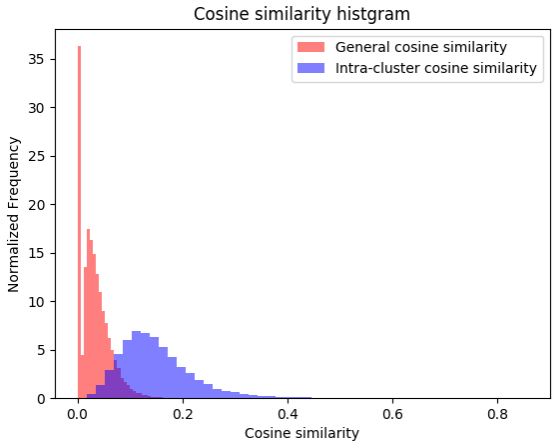

図 1 FAQ 内とランダムに選ばれた文書間の類似度の分布

2.3 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] が使われてきた．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．

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

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 をベースに事前に類似文書のクラスタが所与であることを活かした単語の重みづけ手法，ICDF を開発した．

2.4 NDCG

従来から検索ランキングの評価に NDCG(Normalized Discounted Cumulative Gain)[33] という方法が使われてきた．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．なお，DCG とは，あるクエリに対するランキング結果の良さをランキング対象の文書毎に次の点に基づく有用度を求め，その総和を求めたものである．

- ランキングの上位に来る文書ほど有用である
- 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

NDCG の評価式は以下の通りである．

$$\text{DCG}_p = \sum_{i=1}^{p} \frac{\text{rel}_i}{\log_2(i+1)}$$
$$\text{NDCG}_p = \frac{\text{DCG}_p}{\text{DCG}_{\text{Correct}_p}}$$

ここで rel_i はランキング i で予測された文書に与えられる関連度で，正解ランキングの上位にあるほど高い関連度が付与され，任意のランキング以下の文書には 0 が付与される．DCP_p はランキングの上位 p 件までを評価対象とし，予測された文書の関連の有無をみながら，下位になるにつれて大きなペナルティを課した関連度が足しこまれるようになっている．

3. 問題設定

本稿では，FAQ 化作業支援を目的に頻出する問合せクラスタの各文書の重要度を求め，読むべき文書の順序を決定する．その際，事前に頻出する問合せクラスタは特定された状態からスタートする．入力と出力は次のとおりである．

- 入力：FAQ 化対象となる問合せクラスタ (FAQs) に属する文書
- 出力：クラスタ内で優先的に読むべき文書の順序

4. 提案手法

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

4.1 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，本想定を活かした単語の重み付け方法 ICDF を開発した．本重み付け手法では，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各クラスタでの単語の重み付けを高精度化している．

- 全文書で文書で稀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
- 同一クラスタ内で稀に出現する単語は重要ではない

直感的には，文書全体で重要な単語が同一クラスタ内で多く出現する場合，その単語は非常に重要であり，一方，たとえ文書全体で重要な単語であってもクラスタ内で低頻度に出現する場合は重要でないと見なす．評価式は次のようになる．

$$\text{tf-idf-icdf}_{i,j,c} = \text{tf}_{i,j} \cdot \text{idf}_i \cdot \text{icdf}_{i,c}$$
$$\text{tf}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$
$$\text{idf}_i = \log \frac{|D_{all}|}{|\{d : d \ni t_i \wedge D_{all} \ni d\}|}$$
$$\text{icdf}_{i,c} = \begin{cases} \log \frac{|D_c|}{|D_c| - |\{d : d \ni t_i \wedge D_c \ni d\}| + \alpha} & (|\{d : d \ni t_i \wedge D_c \ni d\}| \geq \alpha) \\ 0 & (\text{otherwise}) \end{cases}$$

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，$|D_{all}|$ は総文書数，$|D_c|$ はクラスタ c に属する文書数，$|\{d : d \ni t_i \wedge D_{all} \ni d\}|$ は総文書中での単語 t_i を含む文書数，$|\{d : d \ni t_i \wedge D_c \ni d\}|$ はクラスタ c 中での単語 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．

例えば，図 2 の word1 は 9 文書中で 2 文書で現れているため IDF 値は高くなるが，Cluster1 の中では 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 内で全ての文書で出現しており，結果的に Cluster1 では word1 より word2 の方が重みが高くなる．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毎

に出現頻度が異なってくる．従って，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．

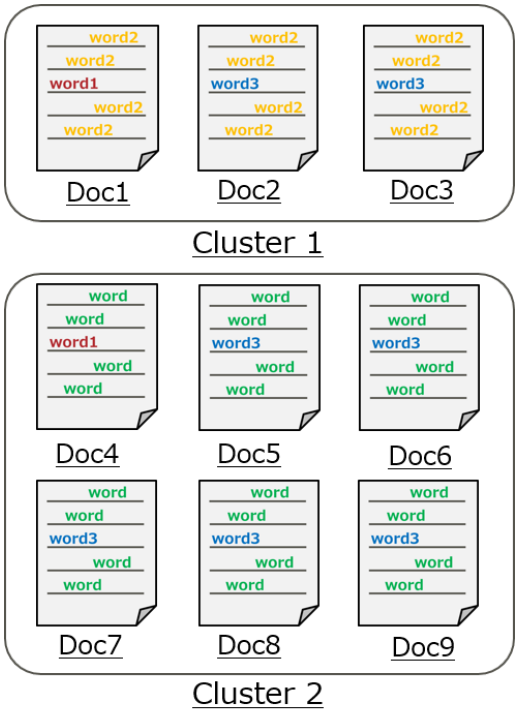

図 2　文書クラスタの例

4.2　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．

$$\mathrm{CTI2}_{c,j} = \sum_{i=1}^{n} \text{tf-idf-icdf}_{i,j,c}$$

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．本手法は，学習なしクラスタ内文書ランキングで活用する．

4.3　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 [34]．例えば，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にあたり，事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．

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

これはクラスタ c の文書 j の単語 t_i の TF-IDF-ICDF が，閾値 β 以上の単語を特徴量として選択する処理を表す．

5.　実験方法

本節では実験方法について説明する．

5.1　実験データの準備

本節では，Stack Overflow から評価データを作成する方法を説明する．本稿では，2018 年 9 月 5 日発行の Stack Exchange Data Dump[7] を用いた．

5.1.1　FAQs の収集

本節では，Stack Overflow から頻出する問合せクラスタを収集する方法を説明する．Stack Overflow では，サイト参加者によって既出の質問に対して Duplicate[19] というタグを付与できるようになっている．例えば，図 3 の例では，プログラミング言語 R[35] で文字列置換する方法に関して 3 つの文書が Duplicate で関連付けれている．このように Duplicate でつながる問合せクラスタは，同種の質問が複数回投稿されたことを示しており，FAQs と言える．

また，Duplicate で関係付けられた 2 文書は重複する内容であっても，Duplicate の関係を多段に辿っていくと，次第に異なる内容になっていることがある．これは複数の FAQs が Duplicate で繋がっている場合に見られる現象で，本稿では対象としない．例えば，図 3 では，全ての文書に“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が記載されている．

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 関係で繋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
- 全ての文書に共通のタグが 1 つ以上付与されていること

今回対象となった問合せクラスタは 35352 個で，166589 文書，2032504 単語で構成されていた．図 4 は今回収集したクラスタサイズの分布である．

5.1.2　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，各投稿に付与された得票数を用いた．Stack Overflow では，投稿を読んだ利用者が役に立った場合は賛成票，そうでない場合は反対票を投じれるようになっており，差分が得票数として付与されている．例えば，図 3 の文書は，それぞれ 238，0，-6 という得票数が付与されている．

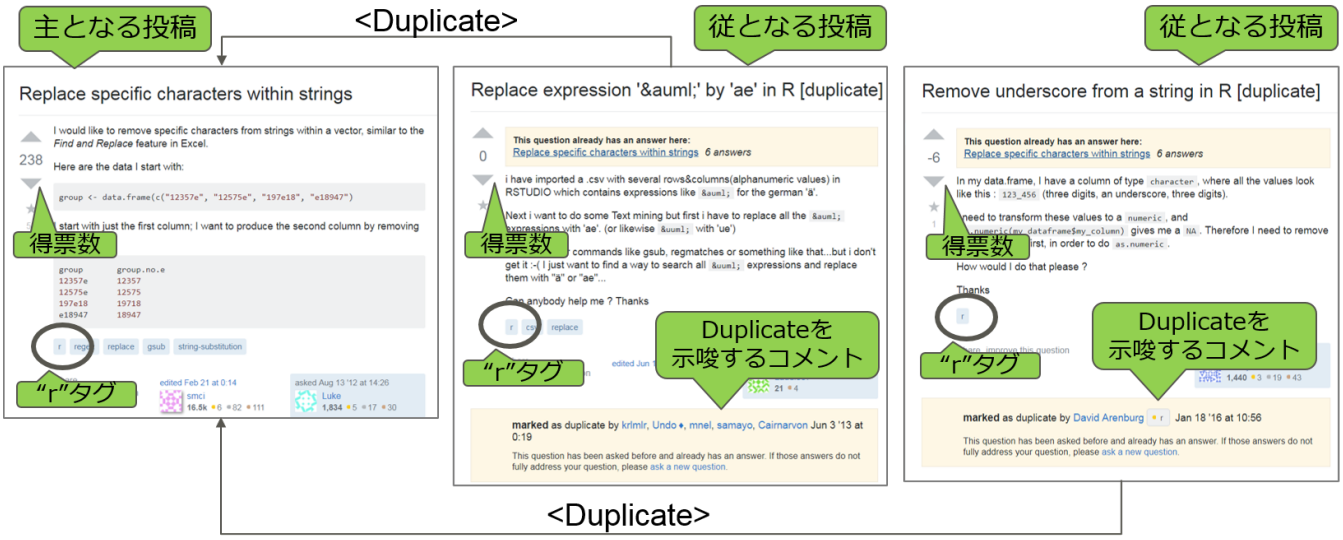

図 3　問合せクラスタの例

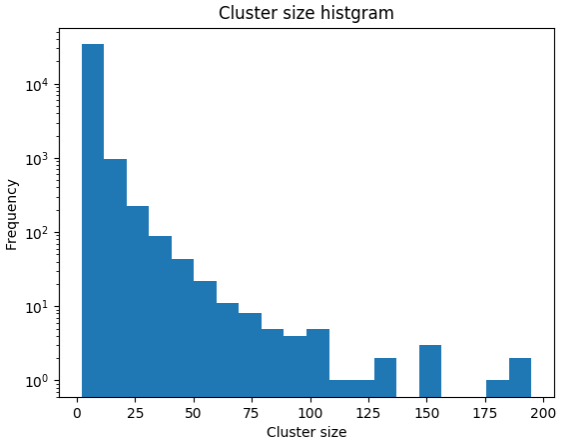

図 4　Stack Overflow から収集した FAQs のサイズの分布

5.1.3　前処理

投稿内容に対する事前処理は以下のとおりである．

1). タイトル，本文，タグの抽出
2). 本文からコード箇所 (<pre><code>と</pre></code>で囲まれた部分) を抽出
3). 本文から Markdown を全て削除
4). タイトル，処理 3 の本文にトークン化する

Stack Overflow の本文は，投稿文を構造的に描画するために Markdown[36] が採用されており，処理 2) や 3) が必要となる．

5.2　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，2.4 節で説明した NDCG(Normalized Discounted Cumulative Gain)[33] 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．

$$\mathrm{rate_d} = |\mathrm{cluster_j}| - \mathrm{rank_d}$$

$$\mathrm{DCG}_{\mathrm{cluster_j}} = \sum_{\mathrm{d} \in \mathrm{cluster_j}} \frac{\mathrm{rate_d}}{\log_2(\mathrm{p\_rank_d} + 1)}$$

$$\text{A-NDCG}_{\mathrm{cluster_j}} = \frac{\mathrm{DCG}_{\mathrm{cluster_j}} - \mathrm{DCG}_{\mathrm{worst_j}}}{\mathrm{DCG}_{\mathrm{best_j}} - \mathrm{DCG}_{\mathrm{worst_j}}}$$

ここで $\mathrm{rank_d}$ 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．$\mathrm{rate_d}$ 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 $\mathrm{rank_d}$ 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，単純化するためである．

$\mathrm{DCG}_{\mathrm{cluster_j}}$ 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）が選択されるためで，$\mathrm{DCG}_{\mathrm{cluster_j}}$ が $\mathrm{DCG}_{\mathrm{worst_j}}$ を下回らないためである．例えば，NDCG では，上位 p 件に全て関連度が 0 の文書が予測されれば NDCG 値は 0 となるが，クラスタ内文書ランキングの $\mathrm{rate_d}$ は，0 以上であり，NDCG 値が 0 になることは決してない．

5.3　クラスタ内文書ランキング

クラスタ内文書ランキングに用いる手法は，学習なしおよび学習あり手法で評価した．また，学習あり手法の入力

特徴量には，(1) テキストメトリクス，(2) 単語の出現ベクトル，BOW(Bag of Words) を用いた．表 1 は実験一覧である．なお，本稿では，クラスタ内の頻度が 1 回，もしくは TF，IDF，ICDF の何れかが 0 の場合は，その単語がノイズであるとみなし特徴量から除外することとした．従って，ICDF のパラメタ α を 1，TI2 Filter のパラメタ β を 0 の場合のみ実験を実施した．

表 1 実験一覧

目的	入力する特徴量	学習なし	学習あり
CTI2 の有効性確認	メトリクス	実験 1	実験 2
TI2 Filter の有効性確認	BOW	—	実験 3

5.3.1 実験 1：学習なしメトリクスベース手法

実験 1 では，テキスト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．

Stack Overflow では，得票数やタグといったユーザからのフィードバック情報を採取可能であるが，通常の Q&A サイトでは同様の情報を利用できるとは限らない．本実験はそのような状況でも高精度にクラスタ内文書ランキングを実施することが可能か確認することを目的としている．

5.3.2 実験 2：学習ありメトリクスベース手法

実験 2 では，教師データを使って，クラスタ内の文書ランキングを予測する，学習ありメトリクスベース手法を評価する．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 [42]（隠れ層のサイズ 10，活性化関数は ReLu[43]）を採用した．このとき学習モデルは教師データのスコアを予測することとした．

5.3.3 実験 3：学習あり BOW ベース手法

実験 3 では，実験 2 と異なり，特徴量に BOW 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した．学習モデルには多層パーセプトロンを用い，構成は図 5 のとおりである．まず入力層に単語数，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た．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．従って，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．

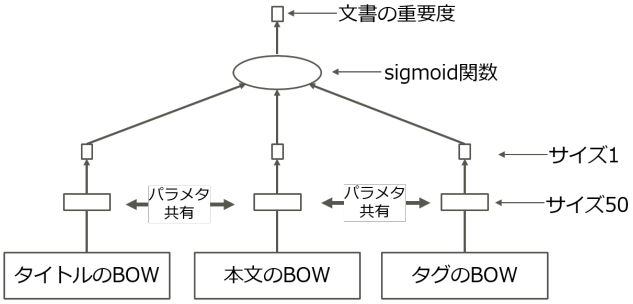

図 5 実験 3 で用いた学習モデル

5.4 実装

実装言語は Python3.5, 機械学習ライブラリは Scikit-learn[44]，Keras[45]，前処理には NLTK[46] を用いた．また，実行環境として，AWS の p2.xlarge[47] を用いた．

6. 実験結果と考察

本節では，実験結果と考察を説明する．

6.1 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．学習なし手法での最高精度は，CTI2 と Tags Count で共に 0.563 あった．また，Cumulative TF-IDF と比較して CTI2 は大きな精度改善が見られ，ICTF を組み込むことによる精度改善が確認できた．

Tags Count が高精度である理由は，高評価を受ける文書ほどより多くのレビューを受け改善されたものが多く，その過程で多くのタグが付与される傾向が強いためである，と推測する．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 は人手に依らず本文から抽出可能なメトリクスであり，適用範囲が広いと言える．

以上より学習なしメトリクスベース手法において，CTI2 は有効なメトリクスであることが確認できた．

6.2 実験 2：学習ありメトリクスベース手法の考察

実験 2 の結果は表 4 のとおりである．単一メトリクスでの手法は，Tags Count や CTI2 を含め特筆する変化が見られなかった．一方，複数のメトリクスにおいては，両モデルを通じて CTI2 による精度改善が見られ，学習なしメトリクスベース手法に対しても精度改善が見られた．以上よ

表 2　学習なし手法で用いるテキストメトリクスの一覧

カテゴリ	メトリクス	説明
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 等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 と解釈した)
	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	質問中の全ての単語の TF-IDF の総和
Readability Metrics	Automated Reading Index[37]	$4.71 \cdot \frac{\text{characters}}{\text{words}} + 0.5 \cdot \frac{\text{words}}{\text{sentences}} - 21.43$
	Coleman Liau Index[38]	$0.588 \cdot \mathrm{L} - 0.296 \cdot \mathrm{S} - 15.8$ 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[39]	$0.39 \cdot \frac{\text{total wotds}}{\text{total sentences}} + 11.8 \cdot \frac{\text{tatal syllables}}{\text{total words}} - 11.9$
	Flesch Reading Ease Score[39]	$206.835 - 1.015 \cdot \frac{\text{total words}}{\text{total sentences}} - 84.6 \cdot \frac{\text{total syllables}}{\text{total words}}$
	Gunning Fog Index[40]	$0.4 \cdot \text{words\_sentences} + 100 \cdot \frac{\text{complex words}}{\text{words}}$
	SMOG Grade[41]	$1.0430 \cdot \sqrt{\text{polysyllables} \cdot \frac{30}{\text{sentences}}} + 3.1291$
Our Method	CTI2(Cumulative TF-IDF-ICDF)	質問中の全ての単語の TF-IDF-ICDF の総和

表 3　実験 1 の結果

カテゴリ	メトリクス	A-NDCG
Textual Metrics	Body Length	0.489
	Emails Count	0.506
	Lowercase Percentage	0.504
	Spaces Count	0.469
	Tags Count	0.563
	Text Speak Count	0.510
	Title Body Similarity	0.518
	Title Length	0.497
	Capital Title	0.529
	Uppercase Percentage	0.510
	URLs Count	0.504
	Code Length	0.463
	LOC Percentage	0.461
	Words Count	0.515
	Sentences count	0.510
	Metric Entropy	0.515
	Cumulative TF-IDF	0.485
Readability Metrics	Automated Reading Index	0.508
	Coleman Liau Index	0.502
	Flesch Kincaid Grade Level	0.507
	Flesch Reading Ease Score	0.509
	Gunning Fog Index	0.515
	SMOG Grade	0.513
Our Method	CTI2($\alpha = 1$)	0.563

り CTI2 が高精度化に寄与すると考えられ，学習ありメトリクスベース手法においても CTI2 の有効性は確認できた．

6.3　実験 3：BOW ベース学習あり手法の結果と考察

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．

- ケース 1：全ての単語を用いる場合
- ケース 2：ランダムに単語を選択した場合
- ケース 3：TF-IDF-ICDF が 0 より大きい値の単語を選択した場合

実験結果は表 5 のとおりである．

表 4　実験 2 の結果

学習モデル	特徴量	A-NDCG
線形回帰モデル	Tags Count	0.563
	CTI2($\alpha = 1$)	0.561
	Textual_Metrics + Readability_Metrics	0.585
	Textual_Metrics + Readability_Metrics + CTI2($\alpha = 1$)	0.587
多層パーセプトロン (隠れ層のサイズ 10)	Tags Count	0.563
	CTI2($\alpha = 1$)	0.561
	Textual_Metrics + Readability_Metrics	0.586
	Textual_Metrics + Readability_Metrics + CTI2($\alpha = 1$)	0.598

表 5　実験 3 の結果

	単語の選び方	入力層	A-NDCG
1	全単語	2,032,504	0.693
2	ランダム	60,549	0.554
3	TI2-Filter ($\alpha = 1, \beta = 0$)	60,549	0.701

実験の結果，ケース 3 において最高精度，次いでケース 1 が高精度であった．また，5.3.3 節で述べたとおり，本実験での計算量は入力層のサイズに比例するため，ケース 1 がケース 3 の約 33 倍の計算量が必要であることが分かる．以上よりケース 3 がケース 1 に比べ大幅なの計算量削減をしつつ精度改善を達成していることが確認できた．一方，ケース 2 はケース 3 と入力層の数は同じで計算量は同等であるが，ケース 2 がケース 3 に対して精度が大きく悪化している．無作為に入力層を削減しただけでは精度を大きく悪化してしまうことが分かる．以上より，TI2 Filiter による適切な特徴量選択によって，計算量削減を行いつつ，精度改善が行えることが分かった．

7. 今後の課題

本節では今後の課題について説明する．

7.1 本手法の改善

本稿では Stack Overflow を使って提案手法の有効性の確認を行ったが，今後，他のデータセットでも同様の有効性を確認したい．また，さらなる高精度化に向けて様々な特徴量の組合せや学習モデルの利用を検討したい．例えば，今回は CTI2 と TI2 Filter を同時に用いる実験は行わなかったが，図 5 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，今回の実験では $\alpha = 1$，$\beta = 0$ の場合のみ実施したが，他の値の組み合わせでの実験も今後検討したい．Yao ら [30] 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していることを示しており，特徴量の参考にしたい．

さらに現実の場面を想定すると，単にランキングするだけでなく，上位何件まで読めば良いか判断できることも重要である．今後，検討を進めていきたい．

7.2 頻出問合せの予兆検知

本稿ではクラウド文書で頻出している内容をソフトウェア文書へ反映することを想定しているが，頻出する前に情報反映しておくことが望ましい．これは予め問合せが頻発しそうなパタンを学習し，問合せが発生した時点で頻出するかどうかを予測するという物である．例えば，OS のセキュリティアップデートの直後は問合せが頻発すると予想しやすい．今回の実験データを活用し，各クラスタの出現頻度を学習できるか検討していきたい．

8. 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，頻出する類似問合せ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前提を活かして，文書クラスタごとに適切な単語の重み付けを行う ICDF (Inverse Complemental Document Frequency) を開発した．またその応用として 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 Score 0.701 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 Proceedings of the Tenth Annual Conference on Computer Assurance Systems Integrity, Software Safety and Process Security'*, IEEE, pp. 273–284 (1995).

[2] Briand, L. C.: Software documentation: how much is enough?, *Seventh European Conference onSoftware Maintenance and Reengineering, 2003. Proceedings.*, IEEE, pp. 13–15 (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, Tech. Rep* (2012).

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

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

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

[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/⟩ （参照）.

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

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

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

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

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

[14] Chaparro, O., Lu, J., Zampetti, F., Moreno, L., Di Penta, M., Marcus, A., Bavota, G. and Ng, V.: Detecting missing information in bug descriptions, *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ACM, pp. 396–407 (2017).

[15] 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).

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

[17] 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)*, IEEE, pp. 563–564 (2017).

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

[19] 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)*, IEEE, pp. 706–716 (2017).

[20] 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)*, IEEE, pp. 307–318 (2015).

[21] Chen, C. and Zhang, K.: Who asked what: Integrating crowdsourced faqs into api documentation, *Companion Proceedings of the 36th International Conference on Software Engineering*, ACM, pp. 456–459 (2014).

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

[23] Henβ, S., Monperrus, M. and Mezini, M.: Semi-automatically extracting FAQs to improve accessibility of software development knowledge, *Proceedings of the 34th International Conference on Software Engineering*, IEEE Press, pp. 793–803 (2012).

[24] Iwai, K., Iida, K., Akiyoshi, M. and Komoda, N.: A help desk support system with filtering and reusing e-mails, *2010 8th IEEE International Conference on Industrial Informatics*, IEEE, pp. 321–325 (2010).

[25] Bozdogan, C. and Zincir-Heywood, N.: Data mining for supporting it management, *2012 IEEE Network Operations and Management Symposium*, IEEE, pp. 1378–1385 (2012).

[26] Bihani, A., Ullman, J. D. and Paepcke, A.: FAQtor: Automatic FAQ generation using online forums, Technical report, Stanford InfoLab.

[27] 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)*, IEEE, pp. 12–23 (2015).

[28] Ponzanelli, L., Mocci, A., Bacchelli, A., Lanza, M. and Fullerton, D.: Improving low quality stack overflow p ost detection, *2014 IEEE International Conference on Software Maintenance and Evolution*, IEEE, pp. 541–544 (2014).

[29] 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, IEEE, pp. 73–82 (2016).

[30] 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).

[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] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N. and Hullender, G. N.: Learning to rank using gradient descent, *Proceedings of the 22nd International Conference on Machine learning (ICML-05)*, pp. 89–96 (2005).

[34] 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).

[35] 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).

[36] Stack Exchange, I.: Markdown help, Stack Exchange, Inc. (online), available from ⟨https://stackoverflow.com/editing-help⟩ (accessed).

[37] Senter, R. and Smith, E. A.: Automated readability index, Technical report, CINCINNATI UNIV OH (1967).

[38] 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).

[39] 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).

[40] Gunning, R.: The technique of clear writing (1952).

[41] Mc Laughlin, G. H.: SMOG grading-a new readability formula, *Journal of reading*, Vol. 12, No. 8, pp. 639–646 (1969).

[42] Bishop, C. M.: *Pattern recognition and machine learning*, springer (2006).

[43] LeCun, Y., Bengio, Y. and Hinton, G.: Deep learning, *nature*, Vol. 521, No. 7553, p. 436 (2015).

[44] 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).

[45] Chollet, F. et al.: Keras (2015).

[46] Bird, S., Klein, E. and Loper, E.: *Natural language processing with Python: analyzing text with the natural language toolkit*, " O'Reilly Media, Inc." (2009).

[47] 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/⟩（参照）.