B-021

タグ推薦機能の既存の文書管理システムへの組み込み手法の設計
An Approach to Incorporate Tag Recommendation in Document Management Systems

蒲沢徹彦 † 瀬野瑛 ‡ 萩原威志 §
Tetuhiko Gawasawa Akira Seno Takeshi Hagiwara

1. はじめに

近年タグを扱うことのできる文書管理システムが一般的になってきた．タグを用いることによってユーザは文書を複数の要素によって分類する事が出来る．タグを使用するとき，文書に対して付けるタグを考えることはユーザにとって手間となる作業である．この手間を軽減しタグを簡単に利用できるようにするため文書管理システムからタグを推薦されることが望ましいが，推薦機能が組み込まれたシステムは一般的でない．そのため，タグ推薦機能が欲しい場合，新たに作り組み込む必要がある．

タグを推薦する手法は今までに様々な手法が提案されている．今回はその中で，文書を解析し推薦するタグを決定する手法を用いる．文書解析を行う手法として潜在意味解析 [1] があり，これを行うために必要なライブラリ [2] が公開されているため，これらを用いることで文書解析に基づく推薦機能を簡単に作成できる．

文書解析に基づく推薦機能を文書管理システムに組み込む場合，通常まず文書をサーバにアップロードし，サーバ側で文書を解析して推薦に必要な情報を抜きだし，この情報を元に推薦するタグを算出することになる．このような手順を踏む場合，アップロード後に算出を行うため，ファイルサイズが大きくなるにしたがってユーザの処理待ちが長くなってしまう．

これらの背景から，本研究ではサーバ・クライアント間でのやりとりを工夫することによってクライアントの処理待ちを減らす手法を提案する．ここで，推薦に必要な情報は文書本体に比べて小さいデータ量であり，推薦するタグを算出するには文書本体は必ずしも必要ないことに着目する．ファイルのアップロード前に推薦に必要な情報のみをサーバ側に送り，推薦するタグの算出を行うことで，ファイル本体をアップロードした後に推薦する場合に比べ短い処理待ちで推薦するタグを提供できる．この方式を文書管理システムに組み込むための設計を行った．

2. 提案手法

組み込む推薦機能は潜在意味解析を利用して作成した．潜在意味解析は自然言語処理で用いられる手法で，文書の比較などに用いられる．潜在意味解析では文書ごとに各単語に特徴量を割り当てた行列に対して解析を行い文書の特徴を算出する．各単語の特徴量として TF-IDF 値 [3] を用いる．TF-IDF 値は単語の重要さを表す指標でその文書での対象の単語の出現頻度と TF-IDF 値算出に使った文書の数，対象の単語の出現した文書の数を用いて算出される．文書中で対象の単語が占める割合が高い程，また，出現した文書が少ない程高くなる．算出した文書の特徴を比較することで文書同士が似ているかどうかを比較する事が出来る．文書の特徴を基にタグの特徴を決めることでタグと文書の特徴を同一の空間で表現する．タグの特徴としてそのタグの付いた文書の特徴の平均を使う．算出したタグの特徴と文書の特徴を比較し，最も似ているものから推薦を行う．潜在意味解析を用いる推薦機能で推薦に必要なデータのうち，クライアントが提供するものは，特徴を算出するために必要となる推薦対象の文書の単語とその出現数である．今後この論文では単語とその出現数を単語/出現数と書く．

提案手法では，文書中の単語/出現数を使って推薦するタグの算出を行う．単語/出現数は文書本体と比べ小さなデータとなるため，クライアント側で単語/出現数のデータを作成しサーバ側に送り，このデータを基に推薦するタグの算出を行う．これによって文書本体をアップロードした後推薦を行う場合に比べ，クライアントの処理待ちを減らすことが出来る．データやりとりの概略を図 1 に示す．

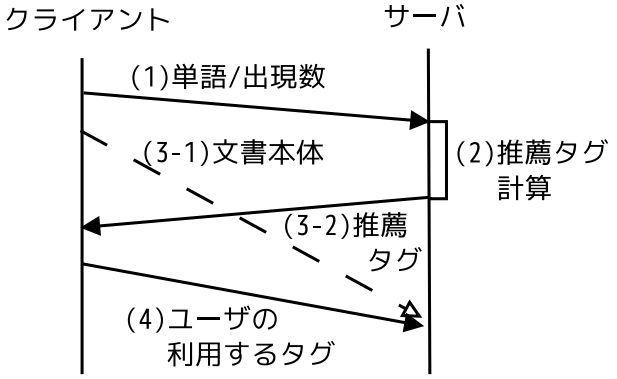

図 1: 文書登録までの処理

以下の手順で行う．(1) クライアントでアップロードする文書から単語/出現数を抽出しサーバへ送る．(2) サーバは単語/出現数のデータを受け取り推薦タグの算出を行う．(3-1) 推薦タグの算出と平行して文書のアップロードを非同期に行う．(3-2) 推薦タグが算出されたらサーバはクライアントに推薦タグを送る．ユーザは推薦されたタグから使うタグを選び新たに追加するタグがあれば入力する．(4) ユーザの利用するタグをサーバへ送る．最終的にサーバは文書とユーザの利用するタグが揃い次第登録を行う．この手順をとることによって文書本体のアップロードと推薦タグの算出の処理を平行して行えるため，ユーザの待ち時間を減らす事が出来る．

3. 実装

提案手法を実現するため推薦機と文書管理システムへ追加する API の設計を行う．独立した推薦機として設計し，文書管理システムの変更は最小限にとどめる．本手法で扱う推薦機能では推薦対象の文書の特徴とそれぞれのタグの特徴を比較して推薦するタグを決めるため，サーバは推薦のための情報としてそれぞれのタグの特徴を持つ必要がある．また，推薦対象文書の特徴をその文書の単語/出現数から算出するために文書管理システムに既に登録されている文書に出現した単語のリストと特徴算出用の情報が必要となる．以後，文書管理システムに既に登録されている文書を登録済み文書と呼ぶ．

それぞれのデータは文書管理システム内の情報であるタグ，タグと登録済み文書の id 間の対応，推薦時に更新する TF-IDF データベースを使って算出する．潜在意味解析はそれぞれの文書，単語に対応する特徴量を使って文書を行，単語を列とした行列を作り特異値分解を使って文書の特徴を算出する．特徴量として TF-IDF 値を使う．本論文では特異値分解に特異値分解ライブラリである red-svd[2] を使った．以下の手順でそれぞれのデータを算出する．

TF-IDF データベースから行列を読み込み特異値分解を行う．また，この時単語リストの更新も行う．特異値分解によって行列は二つの直行行列と一つの対角行列に分解される．二つの直交行列のうち，片方の直行行列の各行は id 順に登録済み文書に対応していて文書の特徴を表すベクトルになる．もう一つの直行行列は推薦対象の文書の単語/出現数から文書の特徴を表す空間での特徴を算出する特徴空間への写像用行列となる．この行列が特徴算出用の情報であり，以後写像用行列と呼ぶ．特異値分解によって分解されたそれぞれの行列は少ない誤差で次元数を削減することができ，次元数は特徴比較の処理時間と精度に影響する．文書の特徴とタグ・文書 id 間の対応から同じタグの付いた文書の特徴を取り出しタグの特徴を算出する．

TF-IDF データベースとタグの特徴算出プログラムを作成した．データ更新の概略を図 2 に示す．

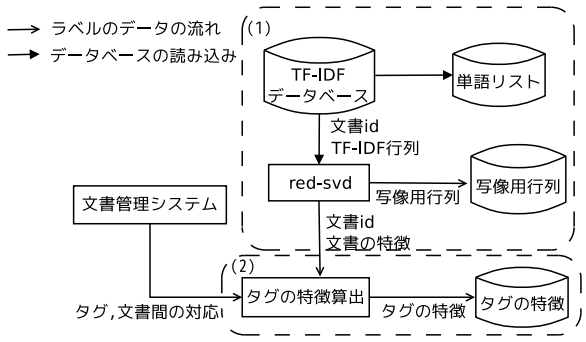

図 2: 推薦情報の算出

矢印がデータの流れを表しラベルは渡されるデータを表す．データベースへの入力が登録である．更新は以下の二つの処理で行う．(1) 文書の特徴算出．TF-IDF データベースから行列を読み込み，同時に単語リストの更新を行う．red-svd に読み込んだ行列を入力して文書の特徴と写像用行列を算出し，写像用行列をデータベースに記録する．(2) タグの特徴算出．文書の特徴とタグ・文書 id 間の対応をタグの特徴算出プログラムの入力としてタグの特徴を算出し，タグの特徴をデータベースに記録する．

推薦時はそれぞれのデータベースの情報と推薦対象文書の単語/出現数を使って推薦するタグを決定する．同時に TF-IDF データベースを更新する．処理の概略を図 3 に示す．

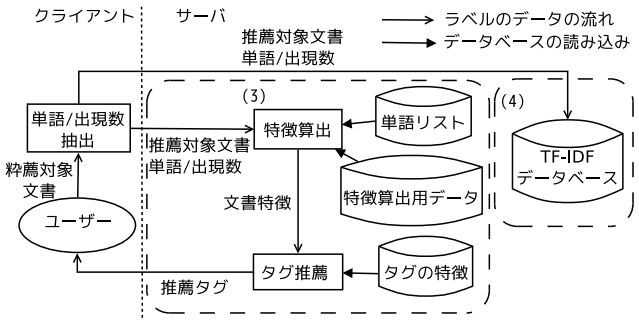

図 3: 推薦時の処理

推薦時には推薦処理と TF-IDF データベース更新の二つの処理を行う．(3) 推薦処理．特徴算出プログラムはクライアントから単語/出現数を受け取り単語リスト，写像用行列を合わせて入力とし推薦対象文書の特徴を算出する．タグ推薦プログラムは推薦対象文書の特徴とタグの特徴を比較し，推薦タグを決定してクライアントに推薦タグを送る．(4)TF-IDF データベース更新．推薦対象文書の単語/出現数を元に TF-IDF データベースを更新する．文書管理システムに追加する API を説明した後，それぞれの処理について述べる．

3.1. 文書管理システムのデータ受け取り用 API

文書管理システムからタグ・登録済み文書 id 間の対応を取得するため，文書管理システムに新たに API を作成する．タグ・登録済み文書 id 間の対応を得るために必要となる API は，タグのリストを出力する全タグ取得 API，入力に応じた文書 id のリストを出力する文書検索 API の二つである．また，登録済みの文書が更新された場合に TF-IDF データベースを更新するため，文書本体を読み出す本文読み出し API を作成する．推薦のために API を追加する必要はなく，タグ推薦プログラムから受け取るタグを表示するようにブラウザへの表示部分を変更するだけでよい．

3.1.1. 全タグ取得 API

タグのベクトルを算出するため，登録されたタグを確認する必要がある．文書管理システムに登録された全てのタグをリストにして出力する．

3.1.2. 文書検索 API

タグの特徴をそのタグのついた文書の特徴を元に算出するため，そのタグのついた文書 id のリストを得る必要がある．タグ，もしくは日付を入力とする．タグを受け取った場合はそのタグの付いた文書の id を出力する．日付を受け取った場合はその日付けから実行日時の間に変更，更新された文書の id を出力する．

3.1.3. 本文読み出し API

登録済み文書 id を入力として対応する文書本体を出力する．

3.2. 文書の特徴算出

TF-IDF データベースから行を文書，列を単語とし，文書中に出現しなかった単語の位置には 0 をいれた行列を読み込む．また，df 値のテーブルから単語のリストを読み込み記録しておく．文書管理システムに登録された文書が m 件，単語リストにある単語が l 種の場合，TF-IDF 行列は表 1 の形式になる．

文書 id	単語 1	単語 2	単語 3	⋯	単語 l
文書 1	0.13	0.17	0	⋯	0
文書 2	0	0.09	0.22	⋯	0
⋮	⋮	⋮	⋮	⋮	⋮
文書 m	0	0	0	⋯	0.02

表 1: TF-IDF 行列

TF-IDF 行列を red-svd の入力にする．red-svd は行列と次元数 n を受け取り特異値分解を行い，特異値分解の結果を n 次元に低減したものを出力する．それぞれの結果は文書の特徴と写像用行列となり，文書の特徴は表 3 の形式の行列になる．

文書 id	特徴量 1	特徴量 2	⋯	特徴量 n
文書 1	0.01	0.004	⋯	-0.01
文書 2	-0.03	-0.01	⋯	-0.002
⋮	⋮	⋮	⋮	⋮
文書 m	-0.005	-0.004	⋯	0.008

表 2: 文書の特徴

3.3. タグの特徴算出

文書の特徴と，タグと登録済み文書 id 間の対応を入力としてタグの特徴を算出する．特徴の比較は cos 類似度を用いるため，タグの特徴はそのタグの付いた文書の特徴の向きの平均とする．文書の特徴の各行をベクトルとみなし，あるタグ $t_i(0 < i)$ の付いた文書を $d_j(0 < j <= m)$ としてタグ t_i のベクトルを以下の式で算出し，その後正規化したものとする．

$$\vec{t_i} = \sum_{j=0}^{m} \frac{\vec{d_j}}{|\vec{d_j}|}$$

文書管理システムに存在する全てのタグについて算出し記録する．

タグ id	特徴量 1	特徴量 2	⋯	特徴量 n
タグ 1	0.01	0.004	⋯	-0.01
タグ 2	-0.03	-0.01	⋯	-0.002
⋮	⋮	⋮	⋮	⋮

表 3: タグの特徴

3.4. 推薦処理

単語/出現数抽出によってクライアント側で推薦対象文書から単語/出現数を抽出してサーバに送る．特徴算出では文書管理システムからクライアントから受け取った単語/出現数を事前に記録した単語リスト，写像用行列と合わせて入力とし推薦対象文書の特徴を算出する．推薦対象文書の特徴は文書の特徴から文書 id を取り除いた形式になる．

特徴量 1	特徴量 2	⋯	特徴量 n
0.01	0.004	⋯	-0.01

表 4: 推薦対象文書の特徴

算出された推薦対象文書の特徴とタグの特徴の cos 比較を行い類似度の高いタグから順に上位 k 件をクライアントにかえす．

3.5.TF-IDF データベース更新

推薦対象文書の単語/出現数から TF 値を算出し TF テーブルに記録していく．また，推薦対象文書のそれぞれの単語について df テーブルに既に存在していればその単語の出現数を 1 増加させ，存在しなければその単語のレコードを作り出現数を 1 とする．TF テーブル，df テーブルはそれぞれ表 5，6 の形式になる．

id	単語	出現数
文書 1	単語 1	0.43
文書 1	単語 2	0.59
文書 2	単語 2	0.29
⋮	⋮	⋮
文書 m	単語 l	0.19

表 5: TF テーブル

4. 評価

実際の文書管理システムに実装を行い，追加する必要のある機能と作業量を調べた．また，推薦時の処理時間に影響を与える次元数について，変動させたとき処理時間の変化を確認した．

文書管理システムへの実装例として OpenKM[4] への実装を行った．OpenKM は OpenKM 社により開発されているオープンソースの文書管理ソフトウェアで

単語	出現数
単語 1	2
単語 2	1
⋮	⋮
単語 l	3

表 6: df テーブル

ある．タグは文書に対してキーワードとしてメタ情報を付与することでその機能をサポートしている．

OpenKM では com.openkm.api に api 群がまとまっているので，今回の実装において必要な api とメソッドをここに追加した．推薦機能を組み込む為に必要となるのは，タグのリストを取得する get_tag_list，任意のタグが付いている文書，もしくは任意の日時以降に変更された文書の id を取得する search_document，任意の id の文書の内容を取得する get_document である．OpenKM においては，既存のクラスとメソッドとして OKMSearch クラスに任意のクエリで登録された文書を検索できる find，keyword(OpenKM ではタグは keyword と呼ばれる) の Map を得る getKeywordMap，keyword で登録済みの文書をリストする findByKeywords がある．また，OKMDocument クラスに任意の文書の id で保存されている path を得るための getPath，指定した path から文書の内容を得る getContent がある．これらを組み合わせてタグのリスト，任意のタグが付いている文書および任意の日時以降に変更された文書を検索し文書の内容を取得することが可能である．これらをラップするメソッドを追加し，受け取る引数を String 型，返す値を List 型，InputStream 型で返すように実装した．OpenKM 5.1.7 に以上の変更を追加した所，get_tag_list で 8 行，search_document_by_tag で 9 行，search_document_by_date で 13 行，get_document で 8 行であった．また，タグを推薦するにあたり，OpenKM のクライアントに推薦機能を利用する為の UI として，推薦されたタグを表示するフォームなどを実装した．

文書の特徴算出時に指定する次元数は小さいほど推薦にかかる処理時間が短くなるが，指定した次元数での近似を得るものであるため，大きな値であるほどもとの特徴に近く正確な推薦が出来ると考えられる．そのため，処理時間が許容できる範囲で収まる次元数を確認する必要がある．推薦時の処理時間を調べるため，推薦処理部分のプログラムのみを動かし処理時間を計る．実験環境は CPU は 3.4GHz，メモリ 35GB である．また，推薦時の処理は単一のスレッドで算出した推薦対象文書の特徴をそれぞれのタグの特徴と順に比較していく方式で行っている．今回は処理時間を 1 秒程度に収めることを目標とする．

調査対象として wikipedia の記事を対象として行った．これは既に多くの記事が揃っており，それらがカテゴリで分類済みであり，wikipedia の各記事は複数のカテゴリに属することが出来るためカテゴリはタグに近い使われ方をしていると言えるためである．登録済み文書として 7934 記事を使った．この中では，単語 181498 種，カテゴリ 12683 種が出現した．これらを使って単語リスト，写像用行列，タグの特徴のデータベースを作成した．推薦機能への入力として経済学の記事を使った．経済学に元から付いているタグは経済，経済学，社会科学であり，推薦タグとして類似度上位 10 件のタグを使った．

次元数を 500 次元から 10 次元の範囲で動かし処理時間の比較と推薦結果の確認を行った．500 次元の場合 9.7 秒，300 次元の場合 6.92 秒，100 次元の場合 2.3 秒，50 次元の場合 1.06 秒，40 次元の場合 0.89 秒，30 次元の場合 0.71 秒，20 次元で 0.45 秒，10 次元で 0.37 秒となった．推薦された結果は 500 次元から 40 次元までは元から付いていたタグが全て推薦された．30 次元から 15 次元では元から付いていたタグのうち一つ以上が推薦された．14 次元以下では一つも推薦されなかった．wikipedia の形式であれば 50 次元程で望ましいタグが推薦され，処理時間を約 1 秒に抑えることが出来る．

5. おわりに

本論文では文書管理システムへのタグ推薦機能の実装方法を提案した．文書本体から推薦に必要となる情報を抽出して事前に送ることで推薦を行い，推薦タグの算出と同時に文書のアップロードを非同期で行うことでユーザの処理待ちを減らすことが出来る．潜在意味解析を用いた推薦機能を提案する方法で実装する場合に必要となる推薦機と文書管理システムに追加する API，個々のプログラムの設計を行った．この設計の実装例として OpenKM に実装を行い，提案する方式でタグ推薦機能を実装するための作業量を確認した．また，潜在意味解析時に算出される特徴の次元数が処理時間と推薦結果に与える影響を調べ処理時間が許容できる範囲に収まる次元数を確認した．

参考文献

[1] LSA @ CU Boulder,
http://lsa.colorado.edu/

[2] japanese - redsvd - RandomizED Singular Value Decomposition ,
http://code.google.com/p/redsvd/wiki/Japanese

[3] 徳永健伸 , 情報検索と 言語処理 , 東京大学出版会

[4] OpenKM Documentation ,
http://www.openkm.com/en/