クラウドソースされた特徴に基づくタスクランキング手法

根本千代之介† 田島 敬史†† 森嶋 厚行 †††

† 筑波大学大学院図書館情報メディア研究科 〒 305–8577 茨城県つくば市天王台 1 丁目 1 番 1 号
†† 京都大学大学院情報学研究科 〒 606–8501 京都府京都市左京区吉田本町
††† 筑波大学 知的コミュニティ基盤研究センター 〒 305–8550 茨城県つくば市春日 1 丁目 2 番地 1 号
E-mail: †s1521637@u.tsukuba.ac.jp, ††tajima@i.kyoto-u.ac.jp, †††mori@slis.tsukuba.ac.jp

あらまし マイクロタスク型クラウドソーシングにおけるタスクの典型として分類がある．分類を行うタスクにおいては，大量のデータ中からある特定のクラスのデータ（正例とよぶ）のみを，少数のタスクで獲得したい場合が存在する．このような場合，タスク中のデータが持っている特徴を利用し，タスクの割り当て順を変更することが有効である．しかし，リクエスタのみで有効な特徴を発見することは必ずしも容易ではない．そこで本論文では，そのための特徴をクラウドソーシングによって獲得する事を提案する．クラウドソーシングによって得られた特徴は玉石混淆であり，その多くが石であるという特徴を持つ．本論文では，クラウドソーシングによって獲得した特徴を選別しながらタスクをランキングする手法として，統計情報を用いた手法と機械学習アルゴリズムを用いた手法をニュース記事を用いたシミュレーションにより比較した．その結果，初期においては，統計情報を用いた手法の方が優位であることが明らかになった．

キーワード クラウドソーシング，分類，タスク割当て制御，特徴選択，ランキング

1. はじめに

クラウドソーシングは多数の人間の力を必要とする業務に適したアプローチであり，様々な場面で用いられている．例えば，自然言語処理における正解データの作成や，画像のタグ付けなどにクラウドソーシングは用いられている．クラウドソーシングの中でも，短時間で処理できる業務を扱うものをマイクロタスク型クラウドソーシングとよぶ．

マイクロタスク型クラウドソーシングにおける典型的な作業の一つとして，データの分類がある．例えば，図 1 に示すタスクは，災害時における被災家屋の発見を目的としたもので，写真の中央にある家屋が倒壊しているか否か分類する．本論文では，このような「短時間でできる，分類を行うタスク」を分類マイクロタスクと呼ぶ．

本論文では，分類マイクロタスクにおいて，大量のデータ中からある特定のクラスのデータのみを少ないタスク数で獲得したい場合に，タスクの割り当て順をどう制御すべきかという問題について議論する．分類マイクロタスクでは，ある特定のクラスのデータのみを少ないタスク数で獲得したい場合が存在する．例えば，航空写真から被災家屋を特定するためのタスク（図 1）を考える．ここでは，支援のために被災家屋を迅速に特定する必要があることから，被災家屋が写っている写真を迅速に獲得できることが望ましい．この場合，「被災家屋が写っている写真」が「ある特定のクラスのデータ」にあたる．なお，以下ではある特定のクラスに属するデータをそのクラスの「正例」とよぶ．

また，上記の場合以外でも，一般に，大量のデータ中からある特定のクラスのデータのみを少ないタスク数で獲得できれば，必要なタスク数を削減でき，ワーカに支払う報酬を削減できる．

図 1 分類マイクロタスクの例

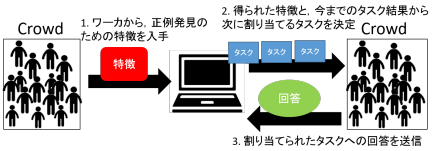

図 2 クラウドソースされた特徴に基づくタスク割り当て制御手法

上に述べたような，少ないタスク数で正例を獲得したい場合には，タスク中のデータが持つ特徴を利用してワーカに割り当てるタスクの順序を変更することが有効である．具体的には，まず，タスク中のデータが持つ特徴からそのデータが正例である可能性を見積もり，その可能性が高い順にデータをランキングする．そしてランキング上位のタスクから先にワーカに割り当てればよい．例えば，図 1 のタスクでは，写真の色の分布やエッジの数といった特徴を利用し，タスクの割り当て順を変更することができる．

しかし，有効な特徴を発見することは必ずしも容易ではない．なぜならば，どの特徴が有効であるかは問題ごとに異なり，また，そもそもデータセットを事前に入手できない等の理由で，あらかじめその性質を知ることは容易ではないからである．実

際，図 1 の被災家屋判定タスクの場合，災害の種類や被災地域の性質によって有効な特徴は変わり，また，災害発生後の被災地域の画像データを災害発生前に取得しておくことはできない．そのため，事前に有効な特徴を発見しておくことは期待できない．また，有効な特徴の中には，それを思いつくのに特定の分野の知識を必要とするものもあり，そうした特徴を発見することは容易ではない．

そこで本論文では，クラウドソーシングによって「有効かもしれない」大量の特徴を入手し，それらの特徴を正例の発見に利用する手法について述べる．クラウドソーシングによって特徴を入手するのは，リクエスタだけでは考えつかない，正例の発見に有効そうな特徴を数多く入手するためである．実際，クラウドソーシングを通して大量の特徴を入手し，その特徴をデータの分類に利用するという研究が存在する [4] [9]．

本手法の全体像を図 2 に示す．本手法では，ワーカから回答を受け取るたびに，今まで得られた全ての回答に基づき特徴を重みづけする．その重みをもとに各タスクのスコアを計算し，スコアが最も高いタスクを次に割り当てる．

第 8. 節で述べるように，クラウドソーシングによって得られた特徴の多くはあまり有効でないという傾向がある．そこで，今まで得られた回答から特徴に重みづけを行う部分が次のように異なる 2 つの手法を比較する．

- 統計情報を利用した重みの決定手法：Statistics based Feature Select（以下では SFSelect とよぶ）この手法では Backward Stepwise Selection [7] と同様に，全ての特徴の中から相対的に有効でない特徴を取り除いていく．ただし，本手法では，重みを 0 と 1 に限定し，有意差の検定を行って相対的に有効でないと考えられる特徴は，直ちに重みを 0 とする．その代わり，重みが 1 である特徴を持つデータが存在しなくなった場合には，一度 0 にした重みを 1 に復活させる．このように，相対的に有効でない特徴をまず捨て，相対的に有効な特徴を優先的に活用するという手法である．
- 機械学習を利用した重みの決定手法：ML

この手法では，ワーカからの回答を訓練データとみなし，既存の機械学習アルゴリズムを用いて各特徴に連続値の重みを付与する．

シミュレーションでは，2 つの手法について，一定数のタスクを処理した時点での再現率と適合率を比較し，それぞれの手法の性質を明らかにする．

本論文の貢献は次のとおりである．

(1) クラウドソーシングによって入手した特徴を用いた学習． 本論文では，学習のための特徴をクラウドソースするという問題を扱う．これまでも，特徴をクラウドソースするという点に着目した研究は存在した [4] [9] が，そこでは，既存のデータの正例と負例を比較して特徴を見つける作業をクラウドソースするというアプローチであった．これは，既に学習のためのデータがあるという事を想定している．本論文では，このような仮定を置かず，クラウドの知識や発想を利用しようとするものである．

(2) クラウドソーシングによって入手した特徴の性質とマッチしたアルゴリズム． 本論文では，玉石混淆であり，かつ，有効でないものがほとんどであるといった大量の特徴が与えられた時に，統計情報を利用した重みの決定手法が，特に初期の段階ではより良い学習を行う事を示した．この手法では，特徴の重みを 1 と 0 に離散化して「すぐに石を捨てる」ことを重視したアルゴリズムを用いている．

(3) 実データを用いた実験． 本論文では，実際のニュース記事のデータと，クラウドソースした特徴を利用した実験を行い，特に初期の段階において，統計情報を利用したタスク割り当て手法が優れていることを確認した．

本論文の構成は以下のようになる．まず，第 2. 節で関連研究を整理する．第 3. 節では特徴のモデル化に用いる特徴量フィルタについて説明する．第 4. 節では本論文で扱う問題を定式化する．第 5. 節では，本論文で述べる 2 つの手法に共通する部分のアルゴリズムについて述べる．第 6. 節では統計情報を利用した重みの決定手法 SFSelect について，第 7. 節では機械学習を用いた重みの決定手法 ML について述べる．第 8. 節ではシミュレーションとその結果を述べる．第 9. 節ではシミュレーションの結果について考察する．第 10. 節では今後の課題を述べ，第 11. 節で本論文をまとめる．

2. 関 連 研 究

本論文は，論文 [10] で提案したアルゴリズムを利用し，実データおよび実際にクラウドソーシングで得られた特徴を用いて行った大規模実験の結果を報告するものである．その他の関連研究は次のとおりである．

多腕バンディット問題 本研究で扱う問題は多腕バンディット問題 [5] と類似している．多腕バンディット問題は知識の「活用」と「探索」がトレードオフの関係にある場合に，一定回数の試行後における利得をいかにして最大化するかという問題である．

本研究で扱う問題とこの問題は，探索と活用のトレードオフが存在するという点で類似している．既に述べたように，少ないタスク数で正例を獲得するには，有効な特徴とそうでない特徴を判別することが必要である．しかし，有効な特徴を判別するために多くのタスクを処理しすぎると，有効な特徴を活用する回数が減り，正例を発見するのに多くのタスクが必要になる．このような性質は，多腕バンディット問題における「活用」と「探索」のトレードオフと類似している．

一方，本研究で扱う問題とこの問題は次の点で異なる．すなわち，一般に多腕バンディット問題では，1 つの試行につき 1 つのアームの性能のみしか調査できないのに対し，本問題では，1 回の試行，すなわち 1 つのタスクが処理されるとき，複数の特徴の良しあしを同時に調査できる．このため，多腕バンディット問題の解法を本研究で扱う問題に直接適用することはできない．

群衆による特徴の提案と特徴量の付与 群衆に特徴の提案と特徴量の付与を行わせた研究として Cheng らの研究 [4] や, Zou らの研究 [9] がある. Cheng らの研究は, ワーカに分類の手がかりとなる特徴を提案させ, ワーカが提案した特徴とリクエスタ自身が考案した特徴を組み合わせた分類器を作成するものであ

る．Zou らの研究は人間にいくつかのデータを比較させることによって，多数の特徴を入手するものである，これらの研究と本研究は，クラウドソーシングによって特徴を入手するという点で類似している．

一方，以下の点が異なる．まず，第 1. 節で述べたように，これらの研究は既存のデータを比較させることで特徴を見つけさせるのに対し，本研究では事前にデータがあることを仮定せず，データ無しで群衆に特徴を提案させる．さらに，Cheng らの研究とは特徴の選別方法が異なる．Cheng らの研究では事前に訓練用データを用意し，それらに対するモデルの精度を比較することで特徴を選別しているのに対し，本研究では事前に訓練データが存在することを仮定せず，ワーカから得られた回答を訓練データと見なし特徴を選別する．また，Zou らの研究とは目的が異なる．Zou らの研究では，あるデータについて様々な種類の特徴を発見することを目的としており，得られた特徴を分類に利用することはしていない．

情報検索における検索結果のランキング作成　本研究で扱う問題は，情報検索における検索結果のランキング作成と類似している．ランキング作成ではランキングの上位に適合文書がくるよう文書を並べ替える．これと同様に，本研究では，正例である可能性が高いタスクから順に処理されるようタスクの割り当て順を決定する．また，本論文で述べる 2 つの手法では relevance feedback [6] と同様の方法を用いて次に割り当てるタスクを決定する．そのため，本論文で述べる手法を情報検索に応用することも可能であり，特に，有効でない特徴が多く含まれる特徴集合を利用しなければならない場合において効果的であると予想される．

3. 特徴量フィルタ

本節では, データが持つ特徴のモデル化に使用する特徴量フィルタ（以降では単に「フィルタ」と呼ぶことがある）を定義する． 特徴量フィルタ f_i とは， データ d_jを引数にとり, d_jが性質 p_i を満たせば 1 を, そうでなければ 0 を返す関数である. 例えば，画像が茶色いか否かを判定するフィルタ $f_{brown}(d_j)$ は, d_j が茶色い画像データであれば 1 を, そうでなければ 0 を返す. 形式的には次のように定義される．

定義 3.1 (特徴量フィルタ)．$D = \{d_1, \ldots, d_n\}$ をデータ集合とし，p_i を判定したい性質とする．このとき，あるデータ $d_j \in D$ が性質 p_i を満たすか否かを判定する特徴量フィルタは，関数 f_i: $D \to \{1, 0\}$ である．

$$f_i(d_j) = \begin{cases} 1 & \text{if } d_j \text{ satisfies } p_i \\ 0 & \text{if } d_j \text{ dose not satisfy } p_i \end{cases}$$

なお，以下では $f_i(d_j) = 1$ のとき，データ d_jはフィルタ f_i にマッチする，とよぶことがある．$f_i(d_j) = 0$ のときはその逆である．

次に空フィルタ f_{emp} を定義する．空フィルタ f_{emp} は全てのデータにマッチする特徴量フィルタであり，次のように定義される．

定義 3.2 (空フィルタ)．空フィルタ $f_{emp} : D \to \{1, 0\}$ は次である．

$$(\forall d_i \in \mathrm{D})\ f_{emp}(d_i) = 1$$

空フィルタは，SFSelect と ML において，最悪の場合でもランダムな順でタスクを割り当てた場合とほぼ同程度の結果となることを保証するために用いられる．

次に，複数の特徴量フィルタを用いて，特徴量ベクトルを定義する．

定義 3.3 (特徴量ベクトル)．性質の列 $p_1 \ldots p_m$ に関するデータ $d_i \in D$ の特徴量ベクトル $\mathbf{x_i}$ とは次である．

$$\mathbf{x_i} = (f_1(d_i), f_2(d_i), \cdots, f_m(d_i))$$

4. 問題定義

本節では，本論文で扱う問題を定式化する．まず，分類したいデータの集合を $D = \{d_1, d_2 \ldots, d_n\}$ とし，D 中の各データがクラス c に属するか判定するタスクの集合を $T = \{t_1, t_2, \ldots, t_n\}$ とする．ここで，各 t_i は d_i がクラス c に所属するか否か分類するためのタスクである．

このとき，本論文における分類タスクの割当て順序制御とは，タスクの処理順序を表す列 T' を動的に作る事である．ここで，T' は T 中の全てのタスクをそれぞれ一度ずつ含む列である．例えば，$T = \{t_1, t_2, t_3\}$ の時，$T' = [t_2, t_1, t_3]$ などが考えられる．

T' を**動的に作る**とは，T' を先頭から順に作る際に，時点 k までに割り当てたタスク列 T'_k (最終的な T'の先頭から k 番目までの接頭辞) に対するタスク結果の集合 $Results$ を見て，次のタスク t_p を決定するという事である．$Results$ の例を表 1 に示す．order はそのタスクが何番目に処理されたかを示す. task_id はタスクの識別子である．また，次に割り当てるタスクを決定する際に，クラウドソーシングによって収集した特徴を利用する．したがって，本論文で扱う手法は，次の手順になる．

（1） 入力として，データ集合 D，タスク集合 T, および特徴の集合 $F_{all} = \{f_1, \ldots, f_m\}$ をとる．

（2） 時点 0 でのタスク列を $T'_0 = []$ とする．

（3） タスクを行ってもらいながら，時点 $k+1$ のタスク列 T'_{k+1} を順次計算する．

（4） $T'(= T'_n)$ と，T' に含まれる分類タスクを処理した結果の集合 $Results$ を出力する．

上記 (3) において T'_{k+1} を求める際の入出力は次の通りである．

入力 ある時点 k までに行われたタスク列 T'_k と，T'_k に含まれる分類タスクを処理した結果の集合 $Results$，および特徴の集合 $F_{all} = \{f_1, \ldots, f_m\}$.

出力 $T'_{k+1} = T'_k + [t_p]$. ただし，t_p は，**特徴を用いて**，$T-$Set (T'_k) から選ばれる．Set は列を集合に変換する関数である．

出力される T' の善し悪しは，いかに少ないタスク数で多くの正例 (クラス c に属するデータ) を発見できたかで評価される．つまり，一定のタスク数がこなされたときの適合率または

表 1　タスク結果の集合 *Results*（4 タスク実施された後の状態）

order	task_id	result
1	10	True
2	2	False
3	4	True
4	9	True

再現率で評価される．

以降では，統計情報を利用した重みの決定手法 SFSelect と，既存の機械学習を利用した重み決定手法 ML を説明する．これらは，上記（3）において，t_p を選ぶために利用する特徴にどう重みづけを行うかの部分が異なる．

5. Feature-based Task Ordering

本節では第 6. 節で述べる SFSelect と第 7. 節で述べる ML に共通する部分のアルゴリズム（Feature-based Task Ordering とよぶ）について述べる．

5.1 概　　要

Feature-based Task Ordering のアルゴリズムを Algorithm1 に示す．このアルゴリズムの入力は タスク集合 T と特徴の集合 F_{all}であり，出力はタスク列 T'である．

このアルゴリズムは次のように動作する．まずタスク処理結果の集合 *Results* と割り当てるタスクの列 T' を初期化する．そして，全てのタスクが処理されるまで次の処理を繰り返す．すなわち，まず，F_{all}と *Results* をもとに各特徴に重みをつける（4 行目）．W は各特徴に対する重みの集合である．この getFeatureWeight 関数の処理が SFSelect と ML で異なる．この関数の処理について第 6. 節と第 7. 節で述べる．そして, 重みづけされた特徴をもとに第 5.2 節で述べる手法で各タスクのスコアを計算し，スコアが最大のタスクを選択する（5 行目）．そして，そのタスクを T' に追加した後，ワーカに割り当て, 得られた結果を *Results* に追加する．なお，複数のタスクが同じスコアを持つ場合，それらの中から割り当てるタスクをランダムに決定する．

5.2 タスクのスコアリング

Algorithm1 中の getTask 関数（5 行目）では，未処理のタスク全てのスコアを計算し，スコアが最も大きいタスクを返す．各タスクのスコアは次のようにして計算する．すなわち，特徴の集合を $F_{all} = \{f_1, \ldots, f_m\}$，それらの重みの集合を $W = \{w_1, \ldots, w_m\}$ とするとき，データ d を分類するタスク t のスコア $score(t)$ を次のようにして計算する．

$$score(t) = w_1 f_1(d) + w_2 f_2(d) + \cdots + w_m f_m(d), \quad w_i \in \mathbb{R}, f_i(d) \in \{0, 1\}$$

例えば，特徴の集合 $\{f_1, f_2, f_3\}$ があり，それらの重みが $(w_1, w_2, w_3) = (2, 1, 1)$ であるとする．また，あるタスク t に対応するデータ d について $(f_1(d), f_2(d), f_3(d)) = (1, 0, 1)$ であるとする．このとき t のスコアは

$$score(t) = w_1 f_1(d) + w_2 f_2(d) + w_3 d_3(d) = 2*1+1*0+1*1 = 3$$

となる．なお，getTask 関数の引数に T と T'があるが，これらは全てのタスクのうちどれが未処理のタスクであるかを判別するために用いられる．

Algorithm 1 Feature-based Task Ordering

Input: T, F_{all}
Output: T'

1: $Results \Leftarrow \emptyset$
2: $T' = []$
3: **while** $|Results|$!= $|T|$ /*全てのタスクが処理されるまでループ*/ **do**
4: W = getFeatureWeights($F_{all}, Results$)
5: $task \Leftarrow$ getTask(F_{all}, W, T, T')
6: T'.add($task$)
7: $result \Leftarrow$ ask($task$) //タスクをワーカに割り当て, 回答を入手
8: $Results$.add($result$)
9: **end while**

6. 統計情報を利用した重みの決定手法：SFSelect

本節では統計情報を利用した重みの決定手法 SFSelect について説明する．SFSelect では統計的検定を利用して，有効そうな特徴に重み 1 を，そうでない特徴に重み 0 を与える．そして重み 1 の特徴を優先して活用する．このように思い切った重みづけを行い，有効ではなさそうな特徴をはやく捨てることで，得られた特徴の中から有効そうな特徴のみをはやく活用するようにしている．重み 1 の特徴にマッチするデータがなくなった場合，重み 0 の特徴間で検定を行い，相対的に有効な特徴の重みを 1 にして活用する．この流れを繰り返す．

6.1 概　　要

SFSelect のアルゴリズムを Algorithm 2 に示す．このアルゴリズムの入力は，特徴の集合 F_{all}と現在までに得られているワーカの回答の集合 *Results* である．出力は F_{all}中の特徴に対する重みの集合 W である．

SFSelect では，特徴の集合 F_{all} 中の各特徴を以下に示す 3 つの特徴集合のいずれか 1 つに所属させる（図 3）．そして，3 つの特徴集合のいずれかに含まれるかによってその特徴の重みを決定する．また，各特徴について，統計的検定を用いて所属させる集合を変更することにより，その特徴の重みを変化させる．

F_{all}中の特徴は $F_{active}, F_{inactive}, F_{finish}$という 3 つの特徴集合のうちのいずれか 1 つに所属する．F_{active} はその時点において有効と判断されている特徴の集合である，また，$F_{inactive}$ はその時点において有効でないと判断されている特徴の集合である．F_{finish}は未処理のタスクのいずれにもマッチしない特徴の集合，つまり，今後活用しても影響を及ぼさない特徴の集合である．F_{active}中の特徴には重み 1 が，$F_{inactive}$および F_{finish} 中の特徴には重み 0 が付与される..

以下では Algorithm 2 を次の 3 つの部分に分けて説明する．

（1） 初期化部分（2 行目〜4 行目）
（2） 特徴の選別部分（6 行目〜20 行目）

（3） 特徴の重みづけ部分（22 行目～29 行目）

6.2 初期化部分

初期化部分では各変数を初期化する．具体的には，次のような処理を行う．まず，F_{active}に F_{all}を代入し，$F_{inactive}$と F_{finish}を空集合とする．つまり，最初は全ての特徴を有効なものであるとみなす．続いて $RTable$ を初期化する．表 2 に示すように，$RTable$ は現在までに処理されたタスクのうち，各特徴にマッチしたタスクが正例であった個数と負例であった個数を保持している表である．$RTable$ はのちに 2 つの特徴の適合率を比較する際に用いられる．

6.3 特徴の選別部分

特徴の選別部分では，各特徴の所属する特徴集合を変更することにより特徴を選別する．具体的には，現在得られているワーカの回答の集合 $Results$ から，処理された順がはやい順に回答を 1 つずつ取り出し次の処理を繰り返す．この処理の結果，各特徴が 3 つの特徴集合のうちのどれに所属するかが決まる．

（1） 回答をもとに $RTable$ の内容を更新する．

（2） F_{active} 中の 2 つの特徴の組み合わせそれぞれについて，$RTable$ をもとに検定を行う（図 3 上部）

（3） （2）の結果，有意差があれば，適合率の劣った方の特徴を F_{active}から $F_{inactive}$に移す（図 3 左下）

（4） F_{all}中の特徴にマッチする未処理のタスクがなくなった場合，F_{all}中の特徴の活用を中止し，$F_{inactive}$中の特徴を活用する（14～18 行目）

（1）では，あるタスクに対するワーカの回答 $result$ が正例であるか否かと，そのタスク中のデータがどの特徴にマッチするか調べる．そして，各特徴について $RTable$ 中の対応するセルの値を 1 増やす（7 行目）．

（2）では F_{active} 中の 2 つの特徴の組み合わせそれぞれについて，適合率に有意差が存在するか $RTable$ を用いて検定を行う．具体的には，10 行目の existSignificantDiff 関数によって，与えられた 2 つの特徴 f_1, f_2 について Precision(f_1) = Precision(f_2) という帰無仮説の検定を行う．なお，仮説検定の手法としては，フィッシャーの正確確率検定とカイ二乗検定を用いた．

（3）では，（2）で有意差が出た場合に，相対的に劣った方の特徴を F_{active}から $F_{inactive}$に移す．つまり，相対的に劣った特徴は有効でないと一時的にみなし，相対的に有効な特徴のみを活用しようとする（12 行目）．

（4）では，F_{active} 中の特徴にマッチする未処理のタスクがなくなった場合，まず F_{active} 中の特徴を全て F_{finish} に入れる（15 行目，図 3 右下）．つまり，使い尽くした特徴を除外する．次に，$F_{inactive}$中の特徴を相対的に有効な特徴とみなし処理を進める（16 行目）．すなわち，現在相対的に有効だとされている特徴を使い尽くしたので，相対的に有効でないと判断されている特徴を利用するようにする．そして，その中で相対的に有効な特徴を活用しようとする．

6.4 特徴の重みづけ

特徴の選別が終了し，各特徴が 3 つの特徴集合のどれに所属するかが確定すると，各特徴の重みが定まる．すなわち，F_{active}

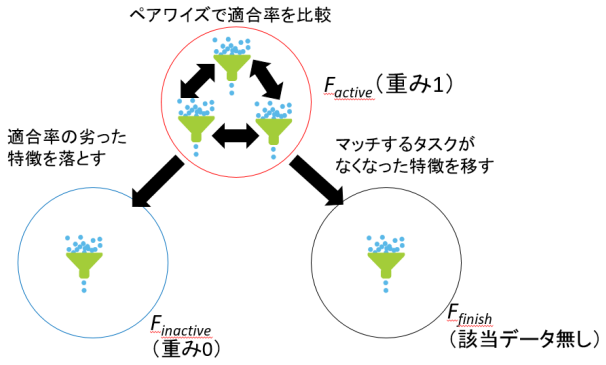

図 3 3 つの特徴集合と特徴選別

表 2 $RTable$ の例

特徴名	正例	負例
f_1	1	10
f_2	8	1
f_3	4	4
f_4	5	2

中の特徴の重みは 1 になる．一方，$F_{inactive}, F_{finish}$ のいずれかに属する特徴の重みは 0 になる．このように，現在相対的に有効だと判断されている特徴の重みのみを 1 とし，その他の特徴の重みを 0 にする．

7. 機械学習を用いた重みの決定手法：ML

本節では，機械学習を用いた重みの決定手法 ML について説明する．この手法ではタスクの処理結果をもとに，機械学習アルゴリズムを用いて各特徴に連続値の重みづけを行う．

ML のアルゴリズムを Algorithm 3 に示す．このアルゴリズムの入力は，特徴の集合 F_{all}と現在までに得られているワーカの回答の集合 $Results$ である．出力は F_{all}中の特徴に対する重みの集合 W である．

このアルゴリズムではワーカの回答の集合 $Results$ を訓練用データとみなす．訓練用データの各インスタンスは第 3. 節で述べた特徴量ベクトルと同じ形式をとる．この訓練用データをもとに，機械学習アルゴリズムを用いて各特徴に対し連続値の重みを付与する．重みを学習するための機械学習アルゴリズムとしてはロジスティック回帰分析を用いた．ロジスティック回帰分析を選択したのは，それが 2 値分類に有効であり広く用いられているためである．なお，訓練用データの量が少なく重みが計算できない場合，その特徴の重みを 0 とした．

8. シミュレーション

「ニュース記事を分類し，テレビに出演するような職業の人の不祥事に言及したニュース記事を少ないタスク数で少数獲得する」というタスクを想定しシミュレーションを行った．このタスクの例を図 4 に示す．まず，使用したデータについて説明する．次に，クラウドソーシングを利用した特徴の入手方法について述べる．そして，SFSelect，ML それぞれについてシミュレーションの結果を述べる．なお，ML については主成分

Algorithm 2 SFSelect

Input: $F_{all}, Results$
Output: W

/*初期化部分*/
$F_{active} \Leftarrow F_{all}$
$F_{inactive} \Leftarrow \emptyset$, $F_{finish} \Leftarrow \emptyset$
initialize($RTable$)
/*特徴の選別部分*/
for each $result$ **in** $Results$ **do**
  updateRTable ($result$)
  for each f_1, f_2 *s.t.* $f_1 \in F_{active} \wedge f_2 \in F_{active} \wedge f_1 \neq f_2$ **do**
    /*各特徴の適合率をペアワイズで比較*/
    if existSignificantDiff($RTable$, $f1$, $f2$) **then**
      $inferiorF \Leftarrow$ getInferiorF(f_1, f_2)
      F_{active}.delete($inferiorF$), $F_{inactive}$.add($inferiorF$)
    end if
    if !(exitstMatchedTask($F_{all}, Results$)) **then**
      $F_{finish} \Leftarrow F_{finish} \cup F_{active}$
      $F_{active} \Leftarrow F_{inactive}$ /*残っている特徴の中で選別を開始する*/
      $F_{inactive} \Leftarrow \emptyset$
    end if
  end for
end for
/*特徴の重みづけ部分*/
for each f_i **in** F_{all} **do**
  if F_{active}.include?(f_i) **then**
    $w_i = 1$
  else
    $/* f_i \in F_{inactive}$ or $f_i \in F_{finished} */$
    $w_i = 0$
  end if
end for
return W

Algorithm 3 ML

Input: $F_{all}, Results$
Output: W

$training\_data = Results$
$W = Classifier.learn(F_{all}, training\_data)$
return W

分析を施した特徴を用いたもの（ML_PCA）と，用いなかったもの両方の結果を示す．また，いずれの手法もランダムな要素の影響を受けるため，各手法について 10 回ずつシミュレーションを行い，その平均を求めた．

8.1 使用したデータ

2016 年の Yahoo!ニュース [1] の記事のうち，「ライフ」，「地域」，「経済」以外のカテゴリに属する記事 6,684 件を用いた．以上の 3 つのカテゴリに属する記事を除外したのは，それらのカテゴリに属するニュースが正例である可能性はかなり低いと考えたためである．

これらの記事全てに対し，上記のテーマに適合しているか否かという 2 値のラベルをクラウドソーシングによって付与した．なお，記事へのラベル付けでは，各記事についてワーカ 3 人にラベル付けを行わせた．3 人の回答が一致した記事についてはその結果をそのまま採用し，回答が割れた記事については，新たに雇用したアノテータ 3 名にラベル付けを行わせ，その 3 人の多数決の結果を採用した．

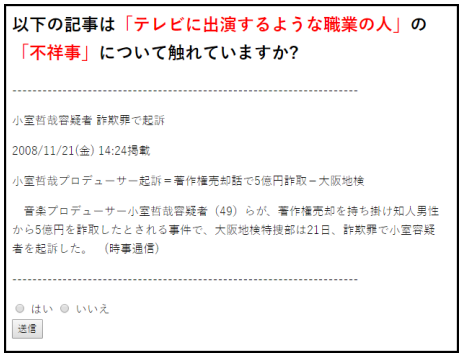

図 4　記事の適合性判定タスクの例

表 3　クラウドソーシングによって得られたキーワード（抜粋）

疑惑
恫喝
事務所
弁明
元人気子役

その結果，適合と判断された記事は 225 件（全体の約 3.8%），不適合と判断された記事は 6,459 件（全体の約 96.2%）であった．また，得られたデータの信頼性を確認するために，得られたデータについて Fleiss の Kappa 係数を求めたところ，その値は 0.667 となった．

8.2 クラウドソーシングによる特徴の入手

Yahoo!クラウドソーシング [2] においてタスクを 200 件投稿し，「テレビ番組に出演するような職業の人の不祥事に関する記事」を検索するためのキーワードを入手した．タスクは自由入力式のものとし，1 タスクにつき 1 個以上のキーワードを入力させた．この結果，キーワードは重複を除いて計 386 件得られた．得られたキーワードの一部を表 3 に示す．

次に，得られたキーワードそれぞれを特徴とみなし，各データに特徴量を付与した．具体的には，正規表現を利用し，各キーワードについて記事にそのキーワードが含まれていれば 1 を，そうでなければ 0 を特徴量として付与した．例えば，「恫喝」というキーワードについての特徴を $f_{恫喝}$ とするとき，記事 d に「恫喝」というキーワードが含まれていれば $f_{恫喝}(d) = 1$ となるようにした．各キーワードから作成した特徴の適合率，再現率を表 4 に示す．なお，得られた特徴のうち，選択率が 0%のものは削除した．なぜなら，選択率が 0%の特徴は削除しても実験結果に影響しないためである．選択率が 0%の特徴を削除した結果，特徴数は 286 となった．この 286 個の特徴をシミュレーションで使用した．

8.3 入手した特徴の性質

クラウドソーシングによって入手した 286 個の特徴の性質を

表 4　シミュレーションで用いる特徴（F 値が高い順に上位 3 件）

	キーワード	選択率	適合率	再現率	F 値
$f_{疑惑}$	疑惑	1.4%	56.2%	21.4%	31.0%
$f_{逮捕}$	逮捕	2.3%	21.7%	13.5%	16.6%
$f_{不倫}$	不倫	0.5%	61.1%	8.7%	15.2%

表 5　各手法での Precision@n

	Random	SFSelect	ML	ML_PCA
Precision@50	3.6%	59.5%	12.8%	33.2%
Precision@100	3.4%	45.0%	22.3%	25.4%
Precision@500	3.6%	26.0%	19.3%	20.4%
Precision@1,000	3.7%	17.7%	14.9%	16.0%
Precision@5,000	3.8%	4.8%	4.9%	4.2%

調べるため，各特徴の適合率をもとにヒストグラムを作成した．それを図 5 に示す．なお，再現率や F 値ではなく適合率をもとにしたのは，少数のタスクで正例を発見するためには適合率が最も重要となると考えたためである．図 5 より，286 個の特徴の大半は適合率が 10%以下である一方で，適合率が 50%を超えている特徴も少数ながら存在していることが分かる．

8.4　主成分分析

ML では，得られた特徴をそのまま用いるものに加え，主成分分析を施し得られた主成分を用いたものについてもシミュレーションを行った．主成分を用いる手法（ML_PCA とよぶ）では，得られた主成分のうち寄与率が高いものから順に 183 個（累積寄与率 81.2%）をシミュレーションに用いた．

8.5　シミュレーションの結果

図 6，図 7 にシミュレーションの結果を示す．図 6，図 7 の横軸は累積タスク数であり，縦軸は再現率である．図 6 は全てのタスクの結果を示したものであり，図 7 は 500 タスク目までの結果のみを示したものである．なお，ランダムな順にタスクを割り当てる手法以外の手法では，特徴発見のために 200 タスクを費やしていることからグラフの描画位置を 200 ずらしている．

また，各手法での Precision@n（n = 100，500，1000，5000）を表 5 に示す．n はその時点までに処理された累積タスク数であり，n が小さいときに Precision@n の値が高い程，少ないタスク数で多くの正例を獲得できていることを示す．

図 6，図 7 より，SFSelect と ML はともにランダムな順にタスクを割り当てる場合（random）と比べて，少ないタスク数で多くの正例を発見していることがわかる．また，SFSelect と ML, ML_PCA では，1,000 タスク目付近までは SFSelect が最も高い再現率となっている一方で，1,000 タスク目以降では ML または ML_PCA が最も高い再現率となっていることが多いことがわかる．また，表 5，図 7 より，SFSelect と ML の差は累積タスク数が少ない時点の方が多い時点と比べ大きいことが確認できる．

9.　考　　察

シミュレーションの結果，タスク数が少ない段階では SFSelect の方が ML よりも有効であった．この理由としては，特徴

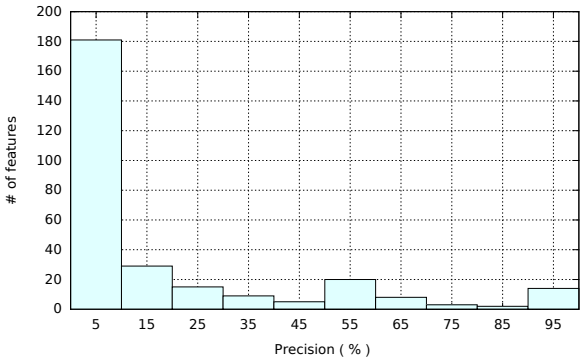

図 5　各特徴の適合率によるヒストグラム

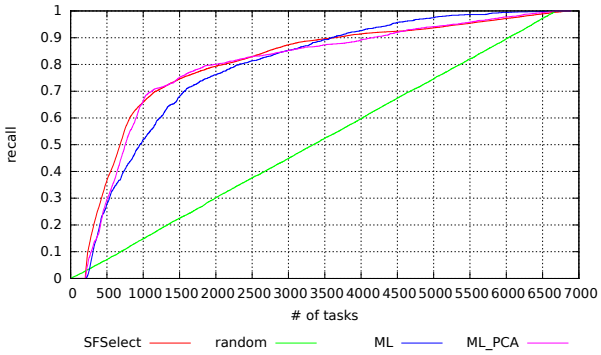

図 6　実施された累積タスク数とそれに伴う再現率の推移（全タスク）

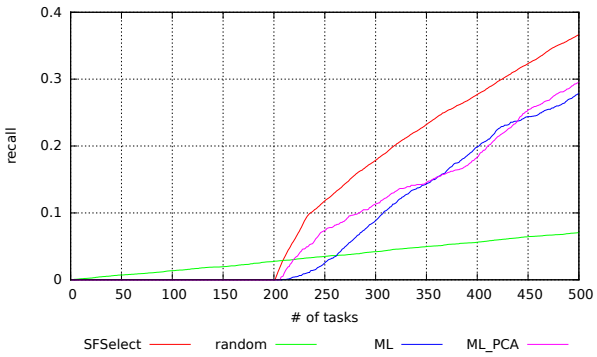

図 7　実施された累積タスク数とそれに伴う再現率の推移（500 タスク目まで）

に対する重みづけ方法の違いが考えられる．ML では各特徴に実数値の重みを付与する一方，SFSelect では 2 値の重みを付与し，有効ではなさそうな特徴の重みをただちに 0 にする．すなわち，SFSelect では，ML に比べ，有効ではなさそうな特徴をはやく捨てる．そのため，クラウドソーシングによって収集した，玉石混淆な特徴を用いる場合，SFSelect の方が少ないタスク数で特徴の選別が行え，多くの正例を発見できたと考えられる．

一方で，タスク数が増えた段階では ML の方が有効となったが，この原因は，ML において学習データが増え，各特徴に適切な重みが付けられるようになったためと予想される．ML では各特徴に実数値の重みを付与するため，離散的な重みを付与する SFSelect よりもより詳しく特徴を選別できる．そのため，学習データが増え，各特徴に適切な重みが付与できるように

なった段階からは，ML の方が良い結果となったと予想される．

10. 今後の課題

今後の課題としては以下の 3 点がある．

（1） シミュレーション結果の要因分析．第 9. 節では，タスク数が少ない段階では SFSelect が ML を上回ることと，タスク数が多くなった段階ではその逆になったことの理由について述べた．しかし現段階では，この理由の妥当性を示すのに十分なデータが得られていない．そのため，さらにデータを収集し，分析する必要がある．具体的には，ML における各特徴の重みの推移や，SFSelect における 3 種類の特徴集合がどう変化しているかについて分析する必要がある．最終稿に分析結果を掲載する予定である．

(2) 他手法との比較．本論文では，ワーカに割り当てるタスクの順序を決定する手法を 2 つ述べた．しかし，ワーカに割り当てるタスクの順序を決定する方法は他にも考えられる．例えば，第 2. 節で述べたように，本論文で扱う問題は多腕バンディット問題と類似していることから，本論文で述べた手法に多腕バンディット問題の解法のアイデアを組み込むことが考えられる．具体的には，特徴を選別する際に，UCB1 [3] のようにそのアームを選択した回数を利用したり，トンプソンサンプリング [8] のように,「アームを引いた際に報酬が得られる確率」の分布を利用することが考えられる．今後は，今回述べた 2 つの手法だけではなく，こういった他手法との比較も行う必要がある．

(3) 統計情報を用いた手法と機械学習アルゴリズムを用いた手法の組み合わせ．第 9. 節で述べたように，処理されたタスク数が少ない段階では統計情報を用いた手法の方が有効であるのに対し，処理されたタスク数が多くなってくると機械学習アルゴリズムを用いた手法の方が有効になっている．このため，タスク数が少ない段階では統計情報を用いた手法を用い，タスク数が増えてきた段階では機械学習アルゴリズムを用いた手法に切り替えることが有効であると考える．この切り替えを行うことにより，序盤だけでなく中盤以降のタスク結果も重要な場合においても，本論文で述べた手法が有効となると考える．ただしその場合，どの程度の数のタスクが処理された時点で手法を切り替えるか決定する必要がある．

11. ま と め

本論文では，大量のデータ中から少ないタスク数で特定のクラスのデータを獲得したい場合に，いかにしてタスクの割り当て順を制御すべきか，という問題を扱い，クラウドソーシングによって入手した特徴を利用したタスク割り当て制御手法を２つ提案した．ニュース記事のデータを用いて実施したシミュレーションの結果，処理されたタスク数が少ない段階では統計情報を利用した手法の方が多くの正例を発見できることが示唆された．今後の課題としては，両手法の切り替えや他手法との比較がある．

謝　　辞

本研究に関してご助言を頂きました筑波大学歳森敦先生，筑波大学若林啓先生，熊本大学櫻井保志先生に感謝します．本研究の一部は，JST CREST および JSPS 科研費 (#25240012) の支援による．

文　　献

[1] http://news.yahoo.co.jp/.
[2] http://crowdsourcing.yahoo.co.jp/.
[3] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, Vol. 47, No. 2-3, pp. 235–256, May 2002.
[4] Justin Cheng and Michael S. Bernstein. Flock: Hybrid crowd-machine learning classifiers. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing, CSCW 2015, Vancouver, BC, Canada, March 14 - 18, 2015*, pp. 600–611, 2015.
[5] T. L. Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, Vol. 6, No. 1, pp. 4–22, 1985.
[6] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.
[7] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning:An Introduction*. MIT Press, 1998.
[8] William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, Vol. 25, No. 3/4, pp. 285–294, 1933.
[9] James Y. Zou, Kamalika Chaudhuri, and Adam Tauman Kalai. Crowdsourcing feature discovery via adaptively chosen comparisons. In *Proceedings of the Third AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2015, November 8-11, 2015, San Diego, California.*, p. 198, 2015.
[10] 根本千代之介, 田島敬史, 森嶋厚行. 分類マイクロタスクにおけるタスク順序制御手法. DEIM2016 第 7 回データ工学と情報マネジメントに関するフォーラム, 2016.