Vol.2020-HCI-190 No.27
Vol.2020-UBI-68 No.27
2020/12/9

機械学習モデルの動的組み合わせによる実世界問い合わせシステム

下田 功一[1,a)]　田谷 昭仁[2,b)]　戸辺　義人[2,c)]

概要：身の回りのセンサによって取得されたデータから，人が見て意味のある情報を得る機会が増えたことで生活の利便性が増している．センサデータから意味のある情報を取得できるサービスは企業が提供するものにとどまらず，個人でも手軽に ML （機械学習）モデルを作り出すことができる環境が整ってきておりウェブ上にも学習済みである ML モデル，いわば予測モデルが多く公開されている．センサデータを入力として用いる予測モデルの例としてスマートフォンの加速度信号から人間の行動を予測するものがある．このように学習済みの ML モデルや企業のサービスは自分で学習をせずに予測に利用することができるが，既存の予測手法では利用者が想定した目的に対し，センサや予測モデルを選択するのみであり，新たな目的やセンサを使用する場合は，その都度，使用するセンサ，予測モデルを選択せねばならない．特に，ML の扱いに慣れていない人にとっては適切に予測モデルを選択することは困難である．あらかじめ登録された ML モデル，信号処理手法，センサの情報から目的に応じてシステムが動的に ML モデルやセンサを組み合わせることができれば利用者は使用できる予測モデルやセンサを登録するだけで予測に利用することができる．本論文では複数の学習済み ML モデル，信号処理手法，センサを集約し，利用者の要求に合わせて動的に組み合わせるシステムを提案，精度評価を行う．複数の分類器を組み合わせて使うことで，単体の予測モデルの使用時より精度が向上することを確認する．

キーワード：機械学習，行動認識，Glove, Word Embedding

1. はじめに

スマートフォンの普及により自宅の様子など遠隔では確認できなかった情報を外出中でも確認できるようになった．生活環境を電子デバイスで確認，操作することの需要の拡大により身の回りのセンサからデータを取得し人々の利用しやすい形で提供するサービスが増加している．また，ML（機械学習）やディープラーニング技術の向上によりカメラ画像から部屋の様子などセンサデータから意味のある情報の取得が可能となった．公開されているデータセットを使って，センサデータを入力とする ML モデルを様々な人が作り出すことができる環境が整ってきている．それにともないウェブ上から学習済み ML モデルをダウンロードし，自身が所持しているセンサと組み合わせることによって手軽に ML を予測に使用することもできるようになった．現在でも，Google Cloud Platform [1] や Microsoft Azure [2] など ML の知識を持たない人々が手軽に利用できるツールが存在する．人々がこれらの学習済み ML モデルやツールを利用するとき，何か一つの目的と使用するセンサがあり，その目的に応じた学習済み ML モデル，サービスを選択して利用する．公開される学習済み ML モデルが増えればその分だけ使用できる選択肢が増えるが，一方で，多数の選択肢の中から使用するモデルを吟味するという作業が必要になる．このような問題に対処するために複数の ML モデルを一括で提供し，利用者が目的に応じて ML モデルを選択しやすくしているサービスも存在する [3]．しかし，既存のサービスは決められた目的のために利用者が選択したセンサを選び，決められた１つの ML モデルの結果を参照するのみであり，利用者の想定する組み合わせのみにしか対応していない．同じ目的のために利用できる ML モデルやその入力となるセンサが複数あった場合，それらを自

[1] 青山学院大学大学院理工学研究科理工学専攻
Graduate School of Science and Engineering, Aoyama Gakuin University
[2] 青山学院大学理工学部情報テクノロジー学科
Department of Integrated Information Technology, Aoyama Gakuin University
[a)] koichi@rcl-aoyama.jp
[b)] taya@it.aoyama.ac.jp
[c)] y.tobe@rcl-aoyama.jp

動で組み合わせるシステムがあれば利用者の知識の範囲を超えた複数のセンサ，ML モデルの組み合わせを提供できる．以上から本研究では複数の学習済み ML モデル，センサを集約し利用者の要求に応じてそれらを動的に組み合わせ提供するプラットフォームを提案する．

2. 関連研究

2.1 Word Embedding モデル

Word Embedding は単語を意味に応じたベクトルに変換する技術である．Word Embedding を獲得する方法として Word2vec [4] や Glove [5] といった手法が存在する．Word2vec は分布仮説に基づきベクトルを求める Skip-gram モデル [6] を用いて学習する手法であり，Skip-gram 手法で学習されたベクトルは単語の関係を表す特徴を持ち，ベクトルの演算を行うことで新たな単語を導き出すことができる．例えば *“king”* − *“man”* + *“woman”* = *“queen”* のように *“king”* から男性成分を引き女性成分を足すことで *“queen”* になる [7]．Glove は Pennington らによって提案された Word Embedding を獲得する手法である．Glove では Skip-gram モデルに共起頻度の比による重み付けを行うことで，より高精度なベクトルが生成される．Skip-gram モデル同様に単語の演算を行うことができる．

2.2 複数 ML を組み合わせて使用する研究

ML モデルを組み合わせる手法として，ある予測に対し，複数の ML モデルの予測結果を説明変数として用いて，正しい予測結果を得るための学習をするスタッキング手法がある．[8] はスタッキング手法を用いて行動認識を行っている．加速度データと音声データを別々のランダムフォレストを用いて学習し Brush teeth など 7 種類の行動の分類を行う．分類の結果，各センサ単体の学習モデルよりも提案されたスタッキング手法が良い結果を示している．スタッキング手法を用いることでの精度向上を見込むことが可能と考えられるが，この手法にも短所がある．スタッキング手法では ML モデルの連結で高い精度が出せているが学習の段階で組み合わせて使う必要がある．そのため，所持しているデータや使用できる ML モデルに応じた柔軟な組み合わせは困難である．本稿は学習済みの ML モデルを組み合わせる手法として Word Embedding を利用して出力ラベルを組み合わせることで精度向上を図る．

2.3 複数センサを組み合わせて使用する研究

複数のセンサを組み合わせて分類を行う研究がある．[9] ではベルトのバックルに搭載したカメラと 3 軸加速度を用いて行動認識を行なっている．[10] は加速度，ジャイロスコープ，地磁気を用いて人の行動認識を行う．[11] は音声とテキストを用いて感情認識を行う．それぞれのデータを単独で使用する場合よりも組み合わせて使用する方が高い精度が得られる．このように複数センサを使用することでの分類精度の向上が報告されている．本研究が提案するシステムでは利用時にアクセス可能なセンサの種類に応じ使用する ML モデルを動的に選択することで，柔軟な対応を可能にする．

3. システムモデル

本章では本システムのシステムモデルを述べる．本システムは利用者が知りたい情報を取得するために ML モデル，信号処理手法，センサ，センサデータ，テキストなどのその他のデータから必要なリソースを動的に選択することで利用者に情報を提供する．この章では ML モデル，信号処理手法，センサ，データを総称してリソースと呼ぶ．

3.1 リソースの定義

リソースの動的な制御を行う管理サーバがあり，遠隔のセンサや ML モデルによる予測を行う計算機と接続されている．ML モデル，信号処理手法，センサなどのリソースは利用者があらかじめ使用できる状態に準備をしておき，これらの利用可能なリソースを管理サーバに登録しておくものとする．リソースは 2 種類に分けることができ，1 つは ML モデル，信号処理手法，センサのように特定の入力に対し，特定の出力をするものでこれらを f_i とする．2 つ目は ML モデル，信号処理手法，センサの入出力となるデータで，データそのものを d_i，データの扱いを決めるための型を D_i とする．f_i が入力および出力するデータをそれぞれ $X_i = (d_{i,1}, d_{i,2}, ..., d_{i,l_i})$，$Y_i = (d_{i,1}, d_{i,2}, ..., d_{i,m_i})$ とすると，

$$Y_i = f_i(X_i), \tag{1}$$

と表される．ここで，l_i，m_i はそれぞれ f_i の入出力するデータの種類の数である．

f_i は決められた型を持つデータのみを入出力とすることができる．入出力のデータの型の集合 $\mathcal{X}_i$，$\mathcal{Y}_i$ をそれぞれ，

$$\mathcal{X}_i = \{D_1, D_2, ...D_{l_i}\}, \tag{2}$$
$$\mathcal{Y}_i = \{D_1, D_2, ...D_{m_i}\}, \tag{3}$$

とするとき，f_i に対して，その入出力の型 $\mathcal{X}_i$，$\mathcal{Y}_i$ をシステムに登録しておく．例として, 画像を入力とし，キャプションを生成する ML モデルを f_a としたとき，入力の型の集合は $\mathcal{X}_a = \{D_{\text{image}}\}$ で出力の型の集合は $\mathcal{Y}_a = \{D_{\text{text}}\}$ となる．加速度とジャイロ信号を入力とし，人間の行動を出力する ML モデル f_b であれば $\mathcal{X}_b = \{D_{\text{acc}}, D_{\text{gyro}}\}$，$\mathcal{Y}_b = \{D_{\text{activity}}\}$ となる．この $\mathcal{Y}_i$，f_i，$\mathcal{X}_i$ の関係を事前に登録しておくことでシステムが使用するリソースの候補として選択することができる．システムに事前に登録された f_i の集合を $\mathcal{F}$ とし，登録された D_i の集合を $\mathcal{D}$ とする．ま

た，センサは入出力とは別にそのセンサの持ち主，設置場所を登録する．このように必要なリソースを準備，設置することによりシステムを使用することが可能となる．

3.2 利用時の動作

システムは利用者の求める情報 $D^\star$ とその条件を入力インタフェイスで受け付ける．入力インタフェイスは 4 つの W（What, Where, When, Who）を利用者に入力してもらう仕組みとなっており，この 4W の値に応じたリソースをシステムが動的に選択する．What には利用者が求める情報 $D^\star$ を入力する．$D^\star$ がシステムに登録されている型の集合 $\mathcal{D}$ に含まれていないときシステムは $D^\star$ を求められずエラーとなる．Where, When, Who は $D^\star$ の条件を入力する項目である．Where は求める情報 $D^\star$ が行われている場所を，When は $D^\star$ が行われている時間を，Who は $D^\star$ の対象となる人物を入力として受け付ける．Where と Who はあらかじめシステムに登録されたセンサの持ち主，設置場所から対象として正しいかが判断され，When はデータが生成された時間から対象として正しいか判断される．例として，shimoda という人物の行動を知りたいとき，What には “Activity” と入力し，Who には “shimoda” と入力することで，得ることができる．roomA という場所に存在する objects を知りたいとき，What には “Objects” と入力し，Where には “roomA” と入力することで，得ることができる．

4. システム設計

本章では提案システムの設計について述べる．複数の学習済み ML モデル，センサ，信号処理手法があるとき，利用者が各手法の知識を持たなければ適切に使用するものを選択することが困難である．そこで，目的に応じて利用者の代わりに ML モデル，センサ，信号処理手法を動的に選択するシステムを設計する．

4.1 ML モデル，センサ，信号処理の動的組み合わせ

データを得るために ML モデル，センサ，信号処理を動的に組み合わせる手法を説明する．3.1 節で定義した f_i をこれ以降ファンクションと呼ぶ．また，センサ f_c は入力 $\mathcal{X}_c = \phi$ で，データ $\mathcal{Y}_c$ を出力する恒等関数とする．アルゴリズム 1 は動的にファンクションを組み合わせるアルゴリズムの疑似コードである．$\mathcal{F}$ に属さないが入力として型 $\{D^\star\}$ を持つ S，ファンクションの組み合わせを保存するためのグラフ構造 $path$ を定義する．$path$ ではファンクションがグラフのノードに相当しデータの型がエッジに相当する．初期値として $\mathcal{F}$ はシステムに登録されているすべてのファンクションの集合，D は求めるデータの型 $D^\star$，$path$ にはグラフのスタートノードとしてファンクションと同様の扱いとなる S を代入する．関数 $SearchPath()$ はファンクション同士の入出力のつながりを探索するための再帰関数で，まず，D を $\mathcal{Y}_i$ の要素として持つ関数 f_i のみを集めた $\mathcal{F}$ の部分集合 $\mathcal{F}'$ を作成する．$path$ のメソッドとして 2 つの引数 S，f の間にあるノードの集合を返す $getNodes()$ を定義する．$\mathcal{F}'$ が空集合ではなかったとき，かつ $path$ 内のノード S から f の間のノードに f_j が属していないとき，$path$ 内のノード f に f_j を結合する．$\mathcal{X}_j$ が空集合でないとき $\mathcal{X}_j$ の要素 D_k を関数 $SearchPath()$ の引数 D に代入し，出力に $D^\star$ が属するファンクションを探索したとき同様，出力に D_k が属するファンクションを関数 $SearchPath()$ を用いて再帰的に探索する．$\mathcal{X}_j$ が空集合 ϕ になるとき，すなわち探索がセンサに到達した場合，または，出力に D が属するファンクションが見つからなかった場合は再帰が行われず，すべてのノードがどちらかの状態に達したとき探索が終了する．このとき完成した $path$ が図 1 のようにファンクション同士をつなぎ合わせる有効グラフとなる．

initialization $\mathcal{F} \leftarrow \mathcal{F}, D \leftarrow D^\star, path \leftarrow S, f \leftarrow$ null
Function *SearchPath*(*F*,*D*,*path*,*f*)**:**
  $\mathcal{F}' \leftarrow \{f_i | f_i \in \mathcal{F}, \mathcal{Y}_i \ni D\}$
  if $\mathcal{F}' \neq \phi$ **then**
    for $f_j \in \mathcal{F}'$ **do**
      if $f_j \notin path.getNodes(S, f)$ **then**
        $path \leftarrow f_j$
        $\mathcal{X}_j \leftarrow$ set of input data type of f_j
        if $\mathcal{X}_j \neq \phi$ **then**
          for $D_k \in \mathcal{X}_j$ **do**
            SearchPath($\mathcal{F}$,D_k,*path*,f_j)
          end
        end
      end
    end
  end
End Function

Algorithm 1: ファンクションのつながりを示すグラフの生成アルゴリズム

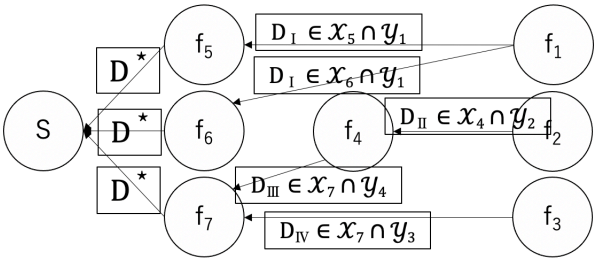

図 1 アルゴリズム 1 により生成されるグラフ

4.2 ファンクションの実行

アルゴリズム 1 で生成されたグラフ $path$ を基にファンクションを順次実行し $D^\star$ 型のデータを取得する．アルゴリズム 2 はアルゴリズム 1 で作成されたグラフ $path$ を基

にファンクションを順次実行していくアルゴリズムである．集合 $\mathcal{P}$ はグラフ $path$ に含まれる全ファンクションの集合である．初期状態としてこの集合 $\mathcal{P}$ から入力が ϕ であるファンクション，すなわち有効グラフにおいてどのノードの終点ともならないセンサ f_i を選択し集合 $\mathcal{P}'$ を作成，データのタプル X，型 $D^\star$ のデータを保存する $Results$ を宣言する．また，集合 $\mathcal{X}_j$ に属する型のデータを一時的に保存する f_j のフィールド $f_j.vars$，d_k の型を返すフィールド $d_k.type$ を定義，$path$ のメソッドとして引数 f_j の子ノードを返す $getChildNode()$ を定義する．関数 $ExFunc$ は引数としてファンクションの集合 $\mathcal{P}'$ とデータのタプル X を持つ．集合 $\mathcal{P}'$ に属する f_j が X と $f_j.vars$ を入力したとき，出力が null でなければ出力を Y に代入する．null の場合，要因として X が f_j に入力できる型ではない，もしくは X だけでは入力に必要なデータすべてを満たしていない場合がある．X だけでは入力に必要なデータすべてを満たしていない場合，$\mathcal{X}_j$ に属するすべての型のデータを同時に入力する必要がある．$ExFunc$ の再帰的な実行により f_j の入力に必要なすべてのデータが揃うまで $f_j.vars$ に X を保存する．$f_j(X, f_j.vars)$ の出力が null でなく，$path$ 内に f_j を親ノードとするファンクションが存在するとき，f_j を親ノードとするファンクションの集合 P'' を第一引数 P'，d_k を第二引数 X として $ExFunc$ を再帰的に実行する．再帰的に実行することにより X を入力したファンクションの出力が子ノードの入力となる．この再帰が繰り返され $path$ 内のファンクションを実行することにより型 $D^\star$ のデータに到達する．d_k が型 $D^\star$ のとき，最終結果を集める $Results$ に格納する．すべての実行可能なファンクションが実行し終わり $Results$ に d_k を格納し終えたとき，アルゴリズムが終了する．

```
initialization P' ← {f_i | f_i ∈ P, X_i = φ}, X ←
  {X_i | f_i ∈ P, X_i = φ}, Results ← null
Function ExFunc(P',X):
    for f_j ∈ P' do
        if f_j(X, f_j.vars) ≠ null then
            Y ← f_j(X, f_j.vars)
            for d_k ∈ Y do
                if d_k.type ≠ D* then
                    P'' ← path.getChildNode(f_j)
                     ExFunc(P'', d_k)
                else
                    Results ← d_k
                end
            end
        else
            if X.type ∈ X_j then
                f_j.vars ← X
            end
        end
    end
End Function
```

Algorithm 2: $path$ 内のファンクションを順に実行するアルゴリズム

4.3 ファンクションの組み合わせによる結果の統合

$ExFunc$ によって複数のファンクションから得られた型 $D^\star$ の複数の変数の値は，認識タスクの場合数値ではなく，自然言語の単語である．認識タスクを行う ML モデルは認識の種類によっては 1 つの値の出力しか行わないものがある．例えば人間の行動認識を行う ML モデルでは通常 1 種類の行動のみを結果として出力する．本システムによって行動認識を行い組み合わされたファンクションから複数の認識結果が得られたとき認識タスクを遂行するには１つの最終結果を出力する必要がある．しかし各ファンクションの認識結果が異なるときに文字列である認識結果を定量的に統合することは困難である．そこで本稿では単語を意味に応じたベクトルに変換することができる Word Embedding を用い単語の演算を行うことで複数の認識結果から 1 つの最終結果を算出する手法を提案する．複数の変数を統合して得られる結果を $label^\star$ とするとき，

$$label^\star = IntegrateLabels(d_1, d_2, ...), \tag{4}$$

のように $label^\star$ は自然言語の単語を統合する関数 $IntegrateLabels$ によって求められる．以下に $label^\star$ を出力する提案手法 $IntegrateLabels$ を説明する．

文字列 d_i を Word Embedding 技術 Glove によりベクトル $\boldsymbol{v}_i$ に変換する関数 $Glove()$ を以下のように定義する．

$$\boldsymbol{v}_i = Glove(d_i), \tag{5}$$

$Glove()$ によって得られたベクトル $\boldsymbol{v}_i$ を以下に示す方法で演算することで最終結果 $label^\star$ を求める．ファンクションが出力する可能性のある単語をラベルと呼ぶ．$label^\star$ を求めるにあたりファンクション f_i が出力するラベル a の重み w_{ia} を以下のように定義する．

$$w_{i,a} = \frac{N_i}{n_a}, \tag{6}$$

ここで変数 N_i はファンクション f_i が出力する可能性のあるラベルの総数，変数 n_a はラベル a を出力する可能性のあるファンクションの総数である．この重みは出力される可能性が少ないラベルの成分が他のラベルの影響により弱くなることを防ぐために用いる．ファンクション f_i が出力するラベル d_i の重み w_{i,d_i} を利用し，重み付けされたベクトルの加重平均ベクトル $\boldsymbol{m}$ は

$$\boldsymbol{m} = \frac{\sum_{i=1}^{n} w_{i,d_i} \boldsymbol{v}_i}{\sum_{i=1}^{n} w_{i,d_i}}, \tag{7}$$

のように表される．ベクトル $\boldsymbol{m}$ とのコサイン類似度 CS を Glove のベクトル空間に存在する全てのベクトル $\boldsymbol{l}$ に対

し計算し，コサイン類似度が最も高くなるベクトル $\boldsymbol{r}$ を選び出す．

$$CS(\boldsymbol{m},\boldsymbol{l}) = \frac{\sum_{j=1}^{|v|} \boldsymbol{m}_j \boldsymbol{l}_j}{\sqrt{\sum_{j=1}^{|v|} \boldsymbol{m}_j}\sqrt{\sum_{j=1}^{|v|} \boldsymbol{l}_j}}, \quad (8)$$

$$\boldsymbol{r} = \arg\max_{l \in L}(CS(\boldsymbol{m},\boldsymbol{l})), \quad (9)$$

$$label = Glove^{-1}(\boldsymbol{r}), \quad (10)$$

ここで $Glove^{-1}()$ は $Glove()$ の逆関数である．このベクトル $\boldsymbol{r}$ を $Glove^{-1}()$ により最終的な結果 $label^{\star}$ に変換する．この結果は Glove のベクトル空間に存在する全ての単語のベクトルから選ばれるので元のファンクションの出力に含まれていない単語が選ばれる可能性もある．

5. 評価

評価としてシステムに複数の $D^{\star}$ を要求し，生成されるグラフを確認，Activity すなわち人間の行動認識の要求に対しファンクションの動的組み合わせと *IntegrateLabels* 手法によりどの程度の精度で予測を行えたかを確認，Activity の要求の際に *IntegrateLabels* 手法により生成された単語の確認を行う．

5.1 評価に用いたファンクション

グラフ生成に用いたファンクションを列挙する．

- f_{LSTM1}：Github にて公開されている [12] の LSTM（long short-term memory）を用いる．学習には [13] のデータセットを用いる．入力の型は加速度データを表す D_{acc}，出力の型は人間の行動認識の結果を表す D_{activity} である． ラベルは “Walking”, “Jogging”, “Upstairs”, “Downstairs”, “Sitting”, “Standing” の 6 つがある．
- f_{LSTM2}：Github にて公開されている [14] の LSTM を用いる．学習には [15] のデータセットを用いる．入力の型は D_{acc} とジャイロを表す D_{gyro}，出力の型は D_{activity} である．ラベルは “WALKING”, “WALKING_UPSTAIRS”, “WALKING_DOWNSTAIRS”, “SITTING”, “STANDING”, “LAYING” の 6 つがある．
- f_{CNN1}：Github にて公開されている [16] の CNN（convolutional neural network）を用いる．学習には [13] のデータセットを用いる．入力の型は D_{acc}，出力の型は D_{activity} である．ラベルは “Walking”, “Jogging”, “Upstairs”, “Downstairs”, “Sitting”, “Standing” の 6 つがある．
- f_{GDL}：Google Vision API の GDL（Google Detect Label）という機能を用いて画像内のラベルを抽出する．ラベルは，一般的なオブジェクト，場所，アクティビティ，動物種，製品などを識別できる．入力の型は画像形式の D_{image} で，出力の型は D_{activity} と人間の感情認識の結果 D_{emotion} である．D_{activity} 型のラベルとして “walking”，“jogging”，“standing”，“sitting” がある．
- f_{HPF}：信号の遮断周波数より低い周波数の成分を逓減させるハイパスフィルタ．本評価では遮断周波数を 0.3 Hz とする．入力の型，出力の型は D_{acc} である．
- f_{TSC}：TSC（Text Sentiment Classifier）[17] は IBM が提供する文章から感情を検出する ML モデル．入力の型は文字のみで構成されるテキストを表す D_{text} で，出力の型は D_{emotion} である．
- f_{IRE}：IRE（Image Resolution Enhancer）[18] は IBM が提供する画像の解像度を 4 倍に増やす ML モデル．入力の型，出力の型は D_{image} である．
- f_{acc}：スマートフォンに搭載されている加速度センサ．出力の型は D_{acc} である．
- f_{gyro}：スマートフォンに搭載されているジャイロセンサ．出力の型は D_{gyro} である．
- f_{cam}：室内に設置されているカメラ．f_{cam1} と f_{cam2} は異なる場所に設置されている．出力の型は D_{emotion} である．
- f_{doc}：テキスト形式の資料が保存されているフォルダ，センサではないが特定の型のデータを出力するので便宜上センサと同じ扱いとする．出力の型は D_{text} である．

5.2 グラフの出力

図 2 は $\mathcal{F}$ に 6 種類の ML モデルと 1 種類のフィルタ，4 種類 5 個のセンサを使用したときの入出力関係を表したグラフである．要求する型として D_{activity}，すなわち人間の行動を選んだとき，赤枠で囲まれているノードで構成されるグラフがシステムにより生成される．4 つの ML モデルにより出力された型 D_{activity} のデータは *IntegrateLabels* により 1 つのラベルとなる．要求する型として D_{emotion}，すなわち人間の感情を選んだとき，緑枠で囲まれているノードで構成されるグラフがシステムにより生成される．2 つの ML モデルにより出力された型 D_{emotion} のデータは *IntegrateLabels* により 1 つのラベルとなる．要求する型として D_{image}，すなわち画像を選んだとき，青枠で囲まれているノードで構成されるグラフがシステムにより生成される．なお，D_{image} を要求するときに使用されるノードはすべて D_{activity}，D_{emotion} 要求時に使用されるノードとなる．

5.3 出力される結果とその妥当性の評価

複数の ML モデル，信号処理を実装し，それらを *IntegrateLabels* で組み合わせたときに得られる結果とその妥当性を評価した．結果の妥当性を評価するために D_{activity} を目的 $D^{\star}$ とするときに組み合わされるファンク

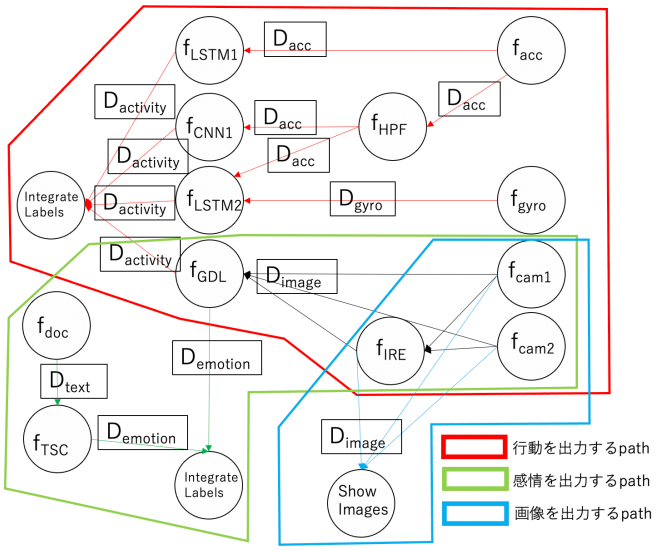

図 2 $\mathcal{F}$ 全体でのグラフと目的に応じた部分グラフ

ションを用いた．人間の行動を記録するための機器としてスマートフォンに内蔵されている加速度計 f_{acc} とジャイロセンサ f_{gyro}，写真を記録するためのビデオカメラ f_{cam1} を用いた．分類に使用するための ML モデルとして f_{LSTM1},f_{LSTM2},f_{CNN1}，f_{GDL}，信号処理手法として f_{HPF} を用いた．静止状態を記録する評価 1 と運動状態を記録する評価 2 を行なったがそれぞれの評価で用いられた f は異なる組み合わせで使用された組み合わせは各項で示す．

5.3.1 Word Embedding モデル

Word Embedding モデルとして Wikipedia で学習された 50 次元，40 万語の Glove モデルを使用した [5]．ただし，“walking_upstairs”，“walking_downstairs” の単語が Glove のベクトル空間上に存在しなかったため “walking_upstairs” の代わりに “walk-up” を採用した．“walking_downstairs” は代わりに該当する単語も存在しなかったため “walk-up” の対として (*“walk”* + *“down”*)/2 の演算結果を 1 つの単語とした “walk_down” という単語を独自に作り出した．

5.3.2 評価 1：静止状態の分類

基本の動きである walking と静止している動きを測るために 1 台のスマートフォンと 1 台のビデオカメラを用いて被験者の walking, standing, sitting, laying の状態を計測した．被験者にはスマートフォンを腹部に装着してもらい，walking, standing, sitting, laying それぞれの状態を 100 秒間持続してもらいビデオカメラでその様子を撮影した．それぞれの状態のデータを ML モデルに入力できるデータとして 33 個ずつ取得した．f は ML モデル $f_{\text{LSTM1}}, f_{\text{LSTM2}}, f_{\text{GDL}}, f_{\text{HPF}}, f_{\text{acc}}, f_{\text{gyro}}, f_{\text{cam}}$ を使用した．$f_{\text{LSTM1}}, f_{\text{LSTM2}}, f_{\text{GDL}}$ 単体での分類と *IntegrateLabels* での分類を行い各正解率を比較した．*IntegrateLabels* における各ラベルの重みは表 1 に示す．

表 1 計測 1 での各ラベルの重み

	ML Models		
	f_{LSTM1}	f_{LSTM2}	f_{GDL}
sitting	2	2	1.33
standing	2	2	1.33
upstairs	3	3	-
downstairs	3	3	-
walking	3	3	-
jogging	3	-	2
laying	-	6	-
running	-	-	4

表 2 計測 2 での各ラベルの重み

	ML Models		
	f_{LSTM1}	f_{LSTM2}	f_{CNN1}
sitting	1	1	1
standing	1	1	1
upstairs	1	1	1
downstairs	1	1	1
walking	1	1	1
jogging	1.5	-	1.5
laying	-	3	-

5.3.3 評価 2：運動状態の分類

動きのあるデータを取得するためにスマートフォン 1 台を用いて被験者の walking, jogging, walking_upstairs, walking_downstairs を計測した．被験者には評価 1 同様スマートフォンを身体に装着してもらいそれぞれの行動を 200 秒間続けてもらった．取得できたそれぞれの状態のデータを ML モデルに入力できるデータとして 45 個ずつ取得した．f は ML モデル $f_{\text{LSTM1}}, f_{\text{LSTM2}}, f_{\text{CNN1}}, f_{\text{HPF}}, f_{\text{acc}}, f_{\text{gyro}}$ を使用した．$f_{\text{LSTM1}}, f_{\text{LSTM2}}, f_{\text{CNN1}}$ 単体での分類と *IntegrateLabels* での分類を行い各正解率を比較した．*IntegrateLabels* における各ラベルの重みは表 2 に示す．

5.3.4 評価 1 の結果

実験結果を表 3 に示す．*IntegrateLabels* は表では IL と略す．まず，各 ML モデルを個別に用いたときの結果では，f_{LSTM1} では “standing” を完全に分類できていたものの “walking”, “sitting” の分類が全くできていなかった．出力することのできないラベルの分類精度ををを 0 とすると全体での分類精度は 25%，出力可能なラベルのみで計算すると 33.3%となった．f_{LSTM2} はすべてのラベルを出力可能で “laying” は 100%の分類精度，“walking” も 90 %越えの精度となったが “standing”, “sitting” では 30%未満の精度となり全体では 57.6%の精度だった．これらの LSTM ではテストデータを用いた際の精度は 90%を超えていたが，被験者の特性などからそれより精度の低い結果となった．f_{GDL} では出力可能なラベル “standing” と “sitting” は 100%の精度で分類できていた．出力できないラベル “walking” と “laying” を含めた全体での結果は 50%となった．*IntegrateLabels* では演算を行うことで最も高い精度と

表 3 評価 1 での分類精度

	f_{LSTM1}	f_{LSTM2}	f_{GDL}	IL
sitting	0/33	2/33	33/33	2/33
standing	33/33	10/33	33/33	33/33
walking	0/33	31/33	-	28/33
laying	-	33/33	-	33/33
total	33/132	76/132	66/132	96/132

表 4 評価 2 での分類精度

	f_{LSTM1}	f_{LSTM2}	f_{CNN1}	IL
upstairs	28/45	0/45	34/45	27/45
downstairs	0/45	16/45	0/45	2/45
walking	13/45	33/45	0/45	21/45
jogging	41/45	-	0/45	41/45
total	82/180	49/180	34/180	91/180

なった．特に “standing” と “laying” は 100%の精度で分類できていた．2 番目に全体での精度が高い f_{LSTM2} と比較すると “standing” での精度の差が最も大きかった．これは *IntegrateLabels* では精度が 100%である f_{LSTM1}，f_{GDL} の結果を反映させることができたからである．また，“laying” の分類精度はラベルの出現しやすさによって重みを付与することにより f_{LSTM2} 同様 100%となった．

5.4 評価 2 の結果

実験結果を表 4 に示す．各 ML モデルを個別に用いたとき，f_{LSTM1} では “jogging” の分類精度が 91.1%となり，ついで “upstairs” の精度も 62.2%となった．反対に “downstairs” の分類精度は 0%となった．f_{LSTM2} では “walking” の分類精度が 73.3%だったが “upstairs” が 0%となった．f_{CNN1} では “upstairs” の分類精度が 75.6%となった．*IntegrateLabels* では重みを用いた演算により “jogging” を f_{LSTM1} 同様の精度で分類することができた．次いで “upstairs” と “walking” の順で精度が高かった．downstairs は 4.44%となった．これはどの ML モデルも downstairs を 50%以上分類できていなかったことが原因である．

5.5 ML モデルの結果とシステムでの結果の比較

各 ML モデルの結果と *IntegrateLabels* 手法を用いて結果を統合したときの出力を表 5 に示す．f_{LSTM1} が “standing”，f_{LSTM2} が “laying”，f_{GDL} が “sitting” を出力したとき *IntegrateLabels* では “laying” となった，この結果は重み付けにより出現頻度の低い “laying” に大きな重みをつけたことで現れた．f_{LSTM1} が “standing”，f_{LSTM2} が “sitting”，f_{GDL} が “standing” を出力したとき多数である “standing” の結果が現れた．f_{LSTM1} が “walk-up”，f_{LSTM2} が “walking”，f_{GDL} が “standing” を出力したときすべて異なる結果から “walking” が現れた．これは，“walk-up” という単語にも walking と同様の歩く意味が含まれており，“standing” と比較しても歩くに上に向かう意味が含まれている “walk-up” よりも “walking” のほうが近いため総合的な意味の近さから “walking” が選ばれたと考えられる．

表 5 各手法での分類結果の比較

f_{LSTM1}	f_{LSTM2}	f_{GDL}	IL	correct answer
standing	laying	sitting	laying	laying
standing	walking	sitting	sitting	sitting
standing	sitting	standing	standing	standing
walk-up	walking	standing	walking	walking

6. 結論

ML モデル，センサ，信号処理を動的に組み合わせ結果を出力するシステムの提案を行った．グラフの生成では，あらかじめ登録されたファンクションから使用できる要素を用いて探索することによって，目的に応じた ML モデル，信号処理，センサの組み合わせを提示できることを確認した．Word Embedding を用いた結果の統合ではそれぞれの ML モデル単体よりも組み合わせることでよい精度を得ることができた．今後，使用できる ML モデルや条件を増やすことでさらなる拡張が期待できる．

参考文献

[1] Google, “Google Cloud Platform.” https://console.cloud.google.com/.
[2] Microsoft, “Microsoft Azure.” https://azure.microsoft.com/.
[3] AXIP, “ailia SDK.” https://ailia.jp.
[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.
[5] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.
[6] C. McCormick, “Word2vec tutorial-the skip-gram model,” 2016.
[7] A. Drozd, A. Gladkova, and S. Matsuoka, “Word embeddings, analogies, and machine learning: Beyond king-man+ woman= queen,” in *Proceedings of coling 2016, the 26th international conference on computational linguistics: Technical papers*, pp. 3519–3530, 2016.
[8] E. Garcia-Ceja, C. E. Galván-Tejada, and R. Brena, “Multi-view stacking for activity recognition with sound and accelerometer data,” *Information Fusion*, vol. 40, pp. 45–56, 2018.
[9] Y. Cho, Y. Nam, Y.-J. Choi, and W.-D. Cho, “Smartbuckle: human activity recognition using a 3-axis accelerometer and a wearable camera,” in *Proceedings of the 2nd International Workshop on Systems and Networking Support for Health Care and Assisted Living Environments*, pp. 1–3, 2008.
[10] H. F. Nweke, Y. W. Teh, G. Mujtaba, U. R. Alo, and M. A. Al-garadi, “Multi-sensor fusion based on multiple classifier systems for human activity identification,” *Human-centric Computing and Information Sciences*, vol. 9, no. 1, p. 34, 2019.
[11] H. Xu, H. Zhang, K. Han, Y. Wang, Y. Peng, and X. Li, “Learning alignment for multimodal emotion recognition

from speech," *arXiv preprint arXiv:1909.05645*, 2019.
[12] "TensorFlow-on-Android-for-Human-Activity-Recognition-with-LSTMs." https://github.com/curiousily/TensorFlow-on-Android-for-Human-Activity-Recognition-with-LSTMs.
[13] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SigKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.
[14] "LSTM-Human-Activity-Recognition." https://github.com/guillaume-chevalier/LSTM-Human-Activity-Recognition.
[15] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones.," in *Esann*, vol. 3, p. 3, 2013.
[16] "Human-Activity-Recognition-using-CNN." https://github.com/aqibsaeed/Human-Activity-Recognition-using-CNN.
[17] IBM, "max-text-sentiment-classifier." https://developer.ibm.com/jp/exchanges/models/all/max-text-sentiment-classifier/.
[18] IBM, "max-image-resolution-enhancer." https://developer.ibm.com/jp/exchanges/models/all/max-image-resolution-enhancer/.