卒業論文

環境認識ライフログからの行動パターン解析による類似性・イベント検出

電子・情報工学科
1415048　福嶋 瑞希
指導教員　奥原　浩之
2018年2月

記号一覧

以下に本論文において用いられる用語と記号の対応表を示す.

用語	記号
クラスター	P，Q
クラスターの重心とクラスター内の各サンプルとの距離の 2 乗和	L(P ∪ Q)
クラスター内での重心とサンプルとの距離の 2 乗和	L(P)，L(Q)
入力データベクトル	x
参照ベクトル	m_i
競合層のニューロンの番号	i
勝者ニューロン	c
勝者ニューロンとの距離によりガウス関数で減衰する係数	h_{ci}
i 番目のニューロンの競合層上での位置	r_i
勝者ニューロンの競合層上での位置	r_c
学習率係数	$\alpha(t)$
学習半径	$\sigma^2(t)$
学習回数	t

目 次

第1章

序論

現代，多くの人がスマートフォンやウェアラブルデバイスを持ち歩くことが一般的であり，急速な情報技術の発達から，個人の生活や行動をデータとして取得，記録することが可能となっている．スマートフォンやウェアラブルデバイスを使用して取得したデータは個人の生活に生かしたり，社会に生かしたりできると考えられている．スマートフォンやウェアラブルデバイスの GPS やカメラ情報をライフログとして取得，解析するアプリケーションが多く存在し，受容性の高いライフログのための研究が行われている．しかし GPS やカメラ映像はライフログデータを取得する本人だけでなく，第三者に対する個人情報が含まれるため，不安や嫌悪感が大きく，手動でライフログデータを取得アプリケーションも多く，未だライフログの受容性は改善の余地がある．多くの人に広く受け入れられるライフログは，個人情報に対する心理的不安，ライフログデータを取得するという物理的負担が少ないものであると考える．ライフログ自体が多くの人に広く受け入れられることで，取得するデータ量を増やすことができるため，より個人や社会に生かすことができると考えられる．したがって，ライフログの在り方は改善すべきであると考える．

本研究は，多くの人に広く受け入れられるライフログとして，個人情報保護に着目し，手間がかからず自動的にライフログデータの取得を行い，取得したデータから類似性やイベント性を考察できることを目的とする．この目的のため，スマートグラスと画像認識を用いたリアルタイム視界情報翻訳アプリケーションを提案する．このアプリケーションを使用したビッグデータ構築，データ解析を行い，行動パターンの類似性・イベント検出を行う．また，このアプリケーションの開発には画像認識 API を使用し，デバイスは EPSON MOVERIOBT-300 を使用する．データの解析は，自己組織化マップ，階層的クラスター分析，多次元尺度構成法（MDS），対応分析，共起ネットワークを行い，読み取り・比較を行うことでライフログデータの類似性やイベント性を考察する．

本論文は次のように構成される．第 2 章では，現状のライフログ・ライフログアプリケーションの問題や特徴，スマートグラスの種類や本研究で使用するスマートグラスについて，視界翻訳を行うための画像認識 API について述べる．次に第 3 章では行動識別や分析手法，分析から考えられる類似性やイベント性について述べる．第 4 章では，本研究の提案手法として開発したアプリケーションについて述べる．第 5 章では開発したアプリケーション

のシミュレーション結果と多変量解析を行ったうえでの行動パターンの類似性・イベント検出について考察を述べる．最後に第6章でまとめと今後の展望を述べる．

第2章

ライフログとスマートグラス

§ 2.1　現状のライフログ

ライフログ（lifelog) とは，人間の活動（life) の記録（log) であり，センサーなどで個人の活動に関するログを取得する行為が，元来のライフログの語源と考えられている．本研究では，この行為をライフログとし，個人の行動履歴に基づいて生み出されるビッグデータをライフログデータと呼ぶこととする．

ライフログデータを取得・活用できるアプリケーションとしてソニーモバイルの Xperia 専用アプリ「Lifelog[1]」がある．このアプリケーションはスマートウェア SmartBand 2 と連携することで歩数や心拍数等のライフログデータを取得し，ユーザ自身が健康管理に生かすことができる．また，自動で位置情報をマップにマッピングできる「マッピング - GPS ログまとめて全部記録[2]」や，手動でマッピングできる「Swarm[3]」というアプリケーションがある．このアプリケーションは行動の記録を取ることができるため，日々の生活や旅行の記録として使用できる．この三つのアプリケーションを中心に，ライフログアプリケーションは全地球測位システム (GPS) を必要とすることが多い．このライフログは自分自身の生活に生かすだけではなく，ビジネスとしてターゲティング広告に生かしたり，学習記録をライフログデータとして取得し，学習に生かしたり [1] できる．

既存の研究として，スマートフォンから得られる位置情報履歴や写真撮影履歴，ツイートを使用したライフログデータから行動特徴抽出・イベント検出を行う研究が行われている [2] [3]．しかし，ライフログデータという個人の活動に関するログから，個人を特定することが安易である場合，個人情報の取り扱いに伴う義務が生じプライバシーの侵害という問題を引き起こす [4]．また，スマートフォンを使用したライフログは，意識的にライフログを使用するため，手間がかかるという問題がある．さらに，ライフログデータとして GPS を使用すると，他人に自分の位置を知られるのではないかという考えが多く，未だユーザーの GPS に対する警戒心が強いのが現状であり，技術面とは異なる課題となっている [5] [6].

[1] http://www.sonymobile.co.jp/myxperia/app/lifelog/
[2] https://play.google.com/store/apps/details?id=org.liteapp.mat2
[3] https://play.google.com/store/apps/details?id=com.foursquare.robin

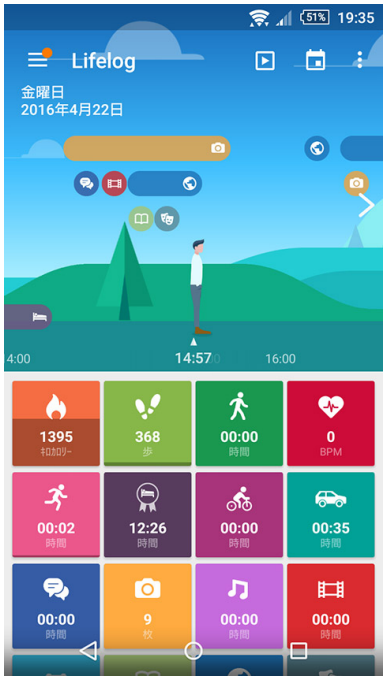

図 2.1: Lifelog

図 2.2: マッピング-GPS ログまとめて全部記録

この問題・課題に対し，ライフログデータを収集する上で，可能な限り不安要素を取り除き，手間をかけず無意識でライフログデータを残すことが重要であると考える．GPS を使用せず，自動でライフログデータを残すことを可能にすることにより，誰でもプライバシー侵害の不安や負担のないライフログを可能にする．

§ 2.2　スマートグラス

近年，スマートフォンやタブレットなどのスマートデバイスが急速に普及している．その次のデバイスとして期待されているものがウェアラブルデバイスである．ウェアラブルデバイスとは，体に装着して利用するコンピュータデバイスの総称であり，スマートグラスはメガネ型のウェアラブルデバイスである．代表的なものとして，Glass Enterprise Edition[4]，SmartEyeglass[5]などが挙げられる．

このようなスマートグラスは把持の必要がなく，常に目の前に仮想画面を表示可能であるため，両手を常に開けておくことが可能である．手をあまり使うことなく欲しい情報を提示することができ，また，外部から見ているものを知られることなく情報を活用できる [7]．さらに，ユーザーが見ている実際の景色に必要な情報を重ねて表示することができるため，拡張現実技術との親和性も高い．

スマートグラスを使用すると，ユーザーがあまり意識する事なく，常時画像の取得を行えるという利点がある．この俊敏性を生かし，ユーザーに負担をかけることなくライフロ

[4]https://www.x.company/glass/
[5]https://developer.sony.com/ja/develop/smarteyeglass-sed-e1/

図 2.3: Glass Enterprise Edition

図 2.4: SmartEyeglass

グデータを取得できる．

本研究では，スマートグラスとしてセイコーエプソン社のシースルーモバイルビューアー MOVERIO BT-300 を使用する (図 2.5)．使用する理由として，スマートグラスの中でも比較的安価であり，Android アプリケーションを作成して動作できるためである．MOVERIO はユーザーが見ている現実空間に対してコンピュータが生成した仮想オブジェクトを重畳表示するというシースルー表示を行う．シースルー表示の実現にはビデオシースルー方式と光学シースルー方式の 2 通りがあり，MOVERIO は光学シースルー方式である．

ビデオシースルー方式は，密閉型の HMD を用いてカメラ画像とコンピューターグラフィックス (CG) を合成した画像を表示する．この方式は，正確な合成ができる反面，ユーザーは直接外界を見ることはできない．これに対し，MOVERIO など光学シースルー方式は肉眼の視界に対して CG を重畳する方式であるため，視界が広く，移動中の使用や現実の物体を用いた作業時の使用に適している．

図 2.5: MOVERIO BT-300

§ 2.3　画像認識 API

画像認識技術とは，コンピュータに画像を理解をさせる技術である．画像内のピクセル信号のパターンから意味を抽出するパターン認識により，人間の視覚機能をコンピュータに処

理させることができる．代表的な画像認識 API に，Google Cloud Vision API と，Computer Vision API という二種類がある．画像認識 API を使用することで，個人だけでは取得が難しい大量のデータを利用することができるため，スマートフォンやウェアラブルデバイスで取得したカメラ画像を認識するアプリケーションの開発が可能となる．

Google Cloud Vision API

2016 年に一般公開された画像認識クラウドサービス Google Cloud Vision API[6]は，写真の被写体を機械判定し，ラベリングする機能をもっている．Google Cloud Vision API を用いて個々の写真の情報（ラベル）を取得できるが，撮影内容が不明瞭のときは，1 つも得られないこともあり [8]，同じ単語を複数個返して来る場合もある．初年度無料である．

Computer Vision API

Microsoft 社が提供している API の一つである Computer Vision API[7]は，写真画像の被写体を機械判読し，ラベリングや画像内のテキストの判読など多様な機能を有している．ラベリングだけでも tags や captions という機能を有する．tags は，画像内の要素を，2,000 以上の認識要素，生物，風景などに基づいて，タグ情報を算出する [9]．captions は文章で人間が読める言語として要約を表示する．初年度無料である．

[6]https://cloud.google. com/vision/
[7]https://azure.microsoft.com/ja-jp/services/cognitive-services/computer-vision/

第3章

行動識別

§ 3.1　行動識別

携帯電話やウェアラブルデバイスを用いて，ユーザーが今何を行っているかという行動をデータとして取得することを行動認識や行動識別という．本研究では，行動識別と呼ぶことにする．

既存研究には，ライフログデータを取得するため，携帯電話の加速度センサや GPS を用いて走行や歩行しているなどの行動を取得する研究 [10] や，ウェアラブルデバイスの加速度センサや GPS を利用する事で人の行うさまざまな行動を識別し取得する [11] 研究がある．しかし，本を読んでいることであったり，料理をしていることなどの細かい動作を取得することは難しい．机上に設置した Kinect(図 3.1) を用いて行動を認識する研究が行われているが，机上に限っているため屋外や机上以外のの行動は認識できない [12]．なお，Kinect は 2017 年 10 月 25 日に生産終了が公表されている．

図 3.1: Kinect

本研究では GPS や Kinect を使用せず，細かい行動を識別する手法として，視界情報の画像認識を行う．ユーザーの視界情報を取得・行動識別を行うと，どこで何をしているのか，誰と会っているのかなど必要以上のライフログデータを取得するため，GPS を使用してい

ないがユーザーのプライバシーを侵害してしまう．さらに，視界情報を画像で蓄積するとデータ量が多くなるという問題が生じる．この問題に対し，ウェアラブルデバイスのカメラ画像を取得し，減色処理を施しデータの蓄積・解析を行う研究が行われている [13]．本研究では画像認識によりカメラ画像をテキストに変換することで，データ量を削減，かつプライバシーに配慮した行動識別を行うことを検討する．

§ 3.2　分析手法

本研究ではいくつかの解析手法を用いて，一定時間内の取得データを視覚的に表す．そのために，多変量解析である自己組織化マップ，階層的クラスター分析，多次元尺度構成法，対応分析，共起ネットワークを用いてテキストデータを可視化を行う．

多変量解析を行うツールとして KH Coder[8]を使用する．KH Coder とは，テキスト型データの計量的な内容分析，もしくはテキストマイニングのためのフリーソフトウェアである [14]．無償でウェブサイトから入手でき、すべての機能をマウス操作で利用できる．また，どんな言葉が多く出現していたのかを頻度表から見ることができたり，自己組織化マップ，共起ネットワークなどの多変量解析を行ったりできる [15]．KH Coder を用いて行われた研究としては，アンケートの自由回答項目・新聞記事・インタビューデータなどさまざまなデータを分析した事例がある [16]．本研究では Version 3.Alpha.11 を使用する．また本章ではチュートリアル用に KH Coder から提供される「坊ちゃん」英語版テキストデータを用いて解析を行う（図 3.2）．

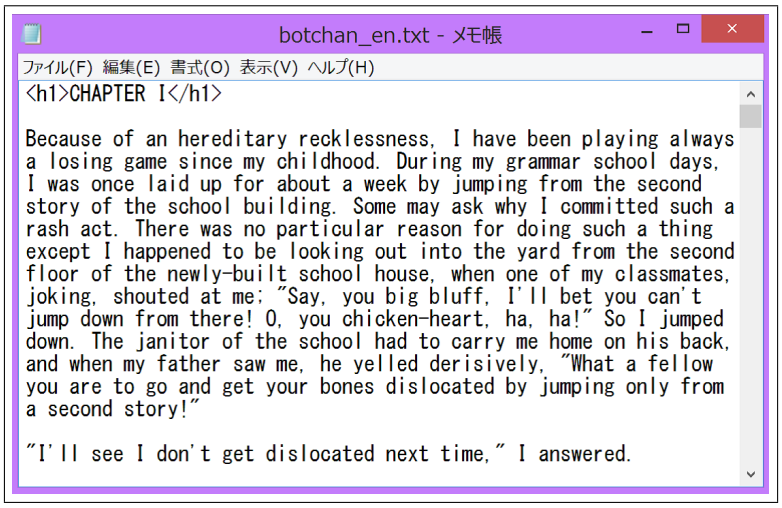

図 3.2:「坊ちゃん」英語版テキストデータの一部

KH Coder を使用した多変量解析には，まず対象のテキストファイル（もしくはエクセルファイル）を読み込む．読み込むテキストファイルに事前に手動で h1 タグや h2 タグという見出しタグを設定することで，見出しごとの解析も可能である．この時 Stop words として Be 動詞のような一般的な語を指定しておくと，分析対象から外すことができる（図 3.3）．

[8]http://khc.sourceforge.net/

次に，前処理として，POS Tagger[9]を使用して自然言語処理を行う（表 3.1）．前処理を行うことで，多変量解析に使用する「各文書に，それぞれの語が何度出現していたのかという集計表」である「文書×抽出語」表 [17](表 3.2) を出力することができる．h1 から h5 というのは見出し番号であり，h1 タグや h2 タグが存在すると 1 増加する．dan は段落番号で，bun は文番号である．id は文書の通し番号で，リセットされることは無い．length_c は文書の長さを文字数で表し，length_w は文書の長さを語数で表したものである．

図 3.3: Stop words の一部

まず，ライフログデータの内容解析のため，階層的クラスター分析，多次元尺度構成法（MDS），対応分析，共起ネットワークを行う．最後に，データの時系列を自己組織化マップ (SOM) を用いて解析を行う．この時テキストデータは，抽出語の中でも，50 回以上出現する 35 語の抽出語を用いる．理由として，出力される抽出語が多すぎると解析結果の読み取りが難しくなるためである．

階層的クラスター分析は，抽出語の最も似ている組み合わせから順番にクラスターにしていく方法であり，デンドログラムを表示する [18]．指定されたクラスター数に全体を分割し，その結果を色分けによって表示する．なお，KH Coder では，デフォルトの Auto では，抽出語数の平方根を四捨五入したものを用いている [19]．1 つのクラスターには互いに関連性が高い語が集まっており、クラスターごとにどういう語が集まっているか調べることで，テキストデータ全体における 1 つの傾向や特徴を読み取られる．作成方法は，まず抽出語として A,B,C,D,E があったとする．この時抽出語の中で最も距離の近い組み合わせ

[9]https://nlp.stanford.edu/software/tagger.html

表 3.1: 抽出語の一部

Noun		ProperNoun	
school	130	Red	171
room	119	Shirt	163
teacher	119	Porcupine	128
house	108	Clown	85
time	95	Kiyo	73
fellow	88	Tokyo	47
day	84	Hubbard	46
student	84	Squash	46
way	78	Badger	32
night	70	Madonna	28
head	65	Koga	23
face	64	Sir	21

表 3.2: 「文書×抽出語」表の一部

h1	h2	h3	h4	h5	dan	id	length_c	length_w	school	room	teacher	house	time
1	0	0	0	0	1	1	650	177	4	0	0	1	0
1	0	0	0	0	2	2	49	16	0	0	0	0	1
1	0	0	0	0	3	3	203	51	0	0	0	0	0
1	0	0	0	0	4	4	43	15	0	0	0	0	0
1	0	0	0	0	5	5	207	58	0	0	0	0	0
1	0	0	0	0	6	6	577	147	1	0	0	1	0
1	0	0	0	0	7	7	853	216	0	0	0	0	0
1	0	0	0	0	8	8	800	193	0	0	0	1	1
1	0	0	0	0	9	9	155	42	0	0	0	0	0

としてAとBとし，AとBをくくる．次にこの2点の代表点を求める．ABの重心，C，D，Eの4点で，最も距離の近い組み合わせを見つける．このときCとDが最も近いとすると，CとDをくくる．このように繰り返していくことで，デンドログラムを作成する．

クラスター間の距離測定方法として，KH Coderではウォード法を使用している．2つのクラスターP，Qを結合したと仮定したとき，それにより移動したクラスターの重心とクラスター内の各サンプルとの距離の2乗和L(P∪ Q)と，もともとの2つのクラスター内での重心とそれぞれのサンプルとの距離の2乗和L(P)，L(Q)の差 (式3.1) が最小となるようなクラスターどうしを結合する手法である．ウォード法は，計算量は多いが分類感度がいいため用いられることが多い．また，ウォード法は一つのクラスターに抽出語が順に吸収され類似するクラスターが形成される鏡効果が起こりにくいという強みがある [20] [21].

$$\Delta = L(P \cup Q) - L(P) - L(Q) \tag{3.1}$$

クラスター分析の結果を図3.4に示す．この時クラスター数をAutoにしたためクラスター数は6となる．併合標準（図3.5）からクラスター数が6であることは妥当だと考えられるためクラスター数は6とした．クラスター分析から，最も多く出現しているsayという単語があるクラスターにschoolという単語がある点から，学校で何かを話す場面が多いのではないかと推測できる．また，teacherとstudentが同じクラスターにある点からやはり学校が重要ではないかと推測できる．クラスター内やクラスターどうしの比較からデータ内で重要な語の関係を推測できる．

多次元尺度法は，抽出語間の関連性や類似性の強さをマップ上の点と点の距離に置き換えて，相対的な関係性を視覚化する手法である [22]．KH CoderではMDSの中でも最も広く利用されてきたKruskalの非計量多次元尺度構成法を使用している [23]．また，語と語の関連を見るためにJaccard係数を使用している．Jaccard係数とは二文章間の類似度であり,「語Aを含む」かつ「語Bを含む」文書の数を,「語Aを含む」または「語Bを含む」どちらかでも当てはまる文書の数で割った係数である (式3.2)．MDSの結果は，相対的な位置関係だけを表わしているため軸の方向性に意味はない．

$$Jaccard\,係数 = \frac{語\,A\,と語\,B\,を含む文書}{語\,A\,もしくは語\,B\,を含む文書} \tag{3.2}$$

取得したデータを使用して出力した多次元尺度法が図3.6になる．クラスター分析を参考にし，クラスター数は6に設定した．この結果，目立つものはクラスター01であり，どのクラスターからも同じような距離であることから，データの中で中心的なクラスター・抽出語であることがわかる．クラスター02と06のように離れて配置されるクラスターもあり，関係性の低い場面であることがわかる．

対応分析は，単純な2次元表や多重表の行と列間の対応する測定値を分析する探索的データ解析の手法であり，分析結果として，2次元のマップが表示される [24]．このマップで近くに位置しているものは，相対的に関連が強いということを示し，遠くに位置しているものは関連が弱いということになる．また，対応分析では，これといって特徴のない語が原点付近に密集することが多い．

取得したデータを使用して出力した対応分析が図3.7になる．この対応分析から，thinkやsayという行動は特徴的ではなく，データ全体によく出現することがわかる．一方で，Red ShirtやClownは原点から遠く離れているため，特徴的であることがわかる．

共起ネットワークはある語が語られる状況の断面を多角的に把握するのに強力なツールであり，線がつながっている語が共起関係にあり，その繋がりにのみ着目する [25]．抽出語の中で出現パターンの似通ったものを線で結ぶ，すなわち共起関係を線で表したネットワークである．多次元尺度法と異なっている点は，プロットされた位置ではなく線で結ばれているかどうかということに意味がある点である．

また，KH Coder では，共起ネットワーク図の表し方として複数用意されており，それらの中から選択できる．まず,「中心性に基づいて色分けが行われており，それぞれの語がネットワーク構造の中でどの程度中心的な役割を果たしているかを表す方法」である．この時の中心性が高い語とは，語がデータ中で重要な役割を果たしている可能性がある語である [26]．次に「比較的強くお互いに結びついている部分を自動的に検出してグループ分けを行い，その結果を色分けによって示すサブグラフ検出方法 [27]」である．この方法は，共起関係の媒介性，random walks および modularity に基づいた方法の中から選ぶことができる．本分析では、各抽出語の関係性を確認するため，KH Coder の出力する共起ネットワーク図の中から「サブグラフ検出（媒介）」を使用する [28]．サブグラフとは，抽出語同士の関係性が強い，つまり Jaccard 係数の値が高い抽出後の集まりであり，KH Coder 特有の言葉であるため，KH Coder のリファレンスマニュアルを参照されたい．

取得したデータを使用して出力した共起ネットワークが図 3.8 になる．この時，Jaccard 係数が 0.2 以上の共起関係を描画している．Jaccard 係数が小さいほど類似度が低いものも含まれ，大きいと類似度が大きいものしか描画されないため状況に応じて検討すべきである．共起ネットワークより，school は teacher や student と共起関係があり，make や say という動詞とも関係性があることがわかる．そのほかの共起関係も視覚的に理解しやすくなっていると考えられる．

SOM は、ヘルシンキ大学のコホーネン教授により 1981 年頃に発表された，教師なし学習を行なうニューラルネットワークの代表例と言える解析手法である [29]．ニューラルネットワークとは、脳機能に見られるいくつかの特性を計算機上のシミュレーションによって表現することを目指した数学モデルである．つまり，人間が無意識にやっていることを機械にやらせるということである．

SOM は図 3.9 に示すように入力層と出力層の 2 つに分かれて競合学習を行う [30] [31]. 今，入力層から出力層への入力を x と定義する．x は入力データベクトルと呼ばれ，1 以上の次元を持つ．図 3.9 には，入力層のニューロンが複数個あるが，各々のニューロンがそれぞれの次元に対応した出力を行っていると考える．

また，出力層のニューロンと入力層のそれぞれのニューロンとの結合強度は総称して参照ベクトルと呼ばれ，m_i で表される．添え字である i は，出力層のニューロンの番号である．ここで，入力ベクトルと各ニューロンの参照ベクトルとのユークリッド距離で出力層のニューロンを競合させる．勝者ニューロンを c とすると，式 3.3 で表される．argminf(x) は f(x) を最小にする x の集合であり，下側に変数がとる値の範囲を書くことが多い．

$$c = \arg\min_i \{||x - m_i||\} \tag{3.3}$$

勝者ニューロンは自らの参照ベクトルと入力ベクトルを近づける学習を行う．また，勝者ニューロンに近いニューロンについても，同じ入力から何かを学習しようと活性化する

ため，参照ベクトルを同様に更新させる (式 3.4).

$$m_i(t+1) = m_i(t) + h_{ci}(t) \cdot \{x(t) - m_i(t)\} \tag{3.4}$$

また，h_{ci} は勝者ニューロンとの距離によりガウス関数で減衰する係数であり，式 3.5 で定義される.

$$h_{ci} = \alpha(t) \cdot \exp \frac{-||r_c - r_i||^2}{2\sigma^2(t)} \tag{3.5}$$

ここで，r_i は i 番目のニューロンの出力層上での位置，r_c は勝者ニューロンの出力層上での位置を示す．また，$\alpha(t)$ は学習率係数，$\sigma^2(t)$ は学習半径という．ともに学習を収束させるため学習回数 t で減衰する係数である．このようなアルゴリズムで SOM は作成される．また，SOM 作成過程ではユークリッド距離を利用している．また，文書の長さのばらつきに左右されない形で計算を行うために，文書中における語の出現回数をそのまま使うのではなく，1,000 語あたりの出現回数に調整したものを計算に使用する．さらに，実行時に距離が 0 となる語の組み合わせがあった場合，自動的にそのうち片方の語が分析から除外される.

自己組織化マップの学習は，大まかな順序づけを行う段階と，微調整を行う収束段階の 2 段階で行われる．KH Coder では，1 段階目が 1,000，2 段階目が「全体のノード数を 500 倍した数値」となっている．また，各ノードがもつベクトルをウォード法で分類してクラスター化する，クラスター数は任意に決定できる.

本研究ではこの KH Coder はデータの前処理段階に使用し，実際の SOM 作成には，データ解析・グラフィックス環境を備えたオープンソースのソフトウェアである R を使用する．取得したデータを使用して出力した SOM が図 3.10 になる．この時 SOM 上の数字は id であり，文書同士の関係が表されている．学習回数は 1 段階目が 1,000，2 段階目が 200,000 となっている．また，クラスター数は 6 とした.

図 3.10 より，文書どうしにまとまりは少ないことがわかる．これは文書数が多いことと，対象としているデータが文学作品であることから似たような文章が並ぶことが少ないためであると考えられる．また，本研究ではライフログデータとして文書の流れを可視化するため SOM を id 順で線を繋いだ SOM（図 3.11）とする．図 3.11 は線を追加することで，似たような文書が続くことが少ないことがわかる.

§ 3.3　類似性・イベント性

KH Coder と R を用いてライフログデータであるテキストデータの多変量解析を行い，解析結果の読み取り・比較を行うことで一定時間内の行動のライフログデータの類似性やイベント性を検出できると考える．本研究では，ライフログデータの類似性とはクラスターの分かれ方やクラスターの構成抽出語が類似していたり，プロットの関係性が類似している場合類似性があると考え，この類似性はユーザー自身の複数のライフログデータの中で類似性のあるライフログデータが多ければ，そのライフログは平常日を表していると考えられる．一方で類似性がなく，イベント性が多いライフログデータであれば，イベント日であることを検出できたり，平常日とは違うという危険を察知したりできる．類似性が検出できると，ユーザー自身がライフログの特徴を知ることができたり，他のユーザーと平

常日の比較を行うことができる．またイベント性を検出できると，平常日とイベント日の比較を行い，平常日だけでは知ることができないユーザーの特徴を検出できる．自己組織化マップはクラスターの分かれ方，クラスターの大きさ，線の流れ方から読み取り・比較を行うことで，多くのライフログの中でも類似したライフログか，特徴的でイベント性のあるライフログかわかる．階層的クラスター分析，共起ネットワークはクラスターの分かれ方やデータを構成する単語の関係性からどのような行動があるのかがわかる．多次元尺度構成法，対応分析はプロット点やプロット間隔から類似する行動やイベント性のある行動がわかる．

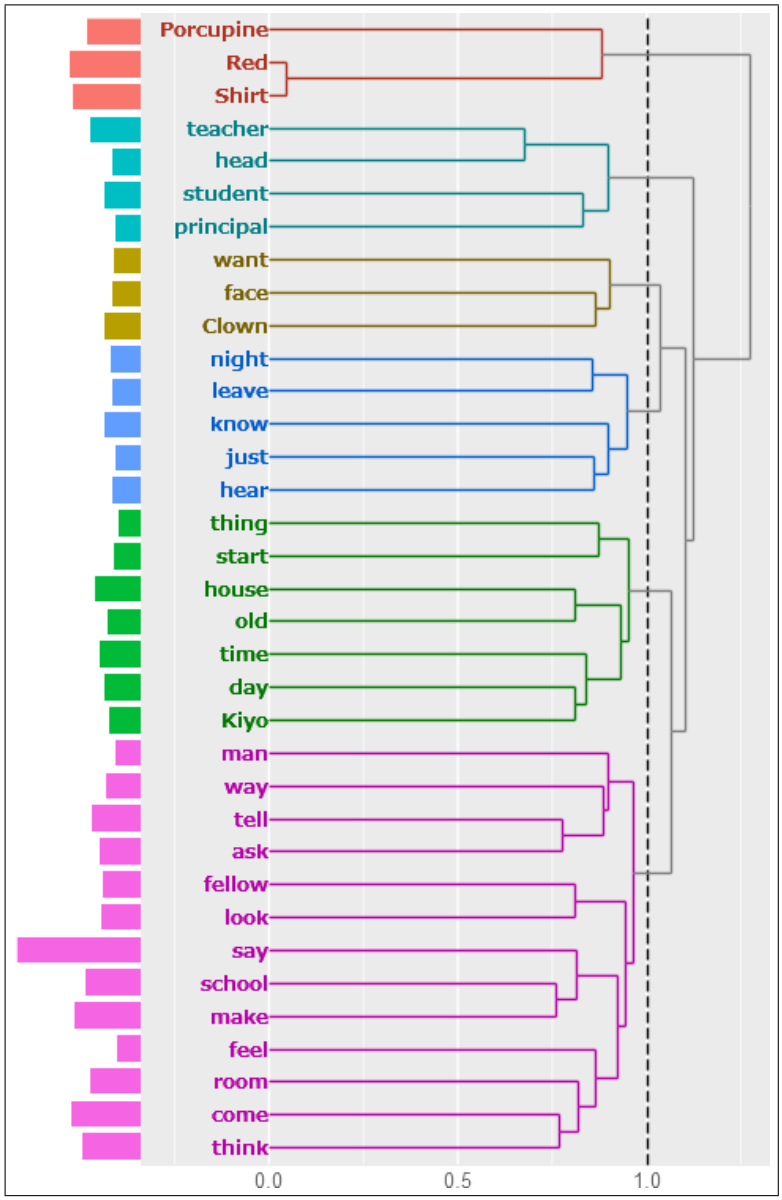

図 3.4: データから作成したクラスター分析

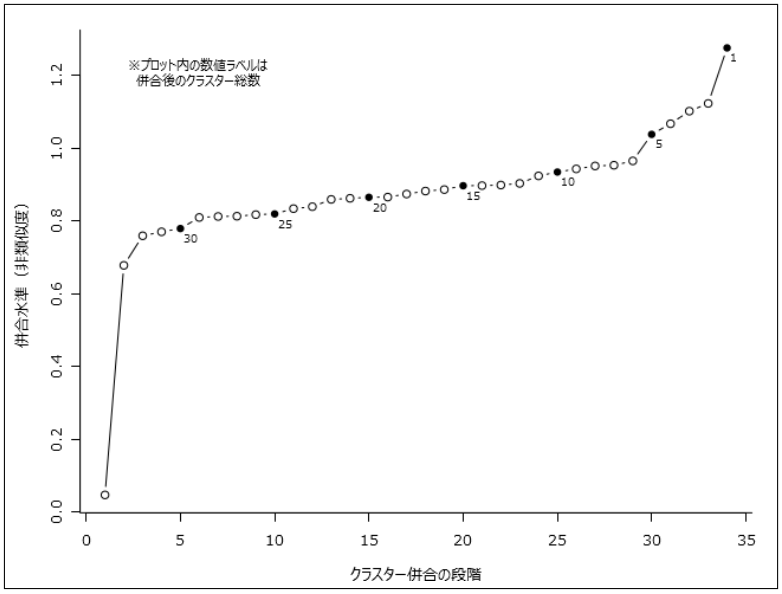

図 3.5: クラスター分析の併合標準

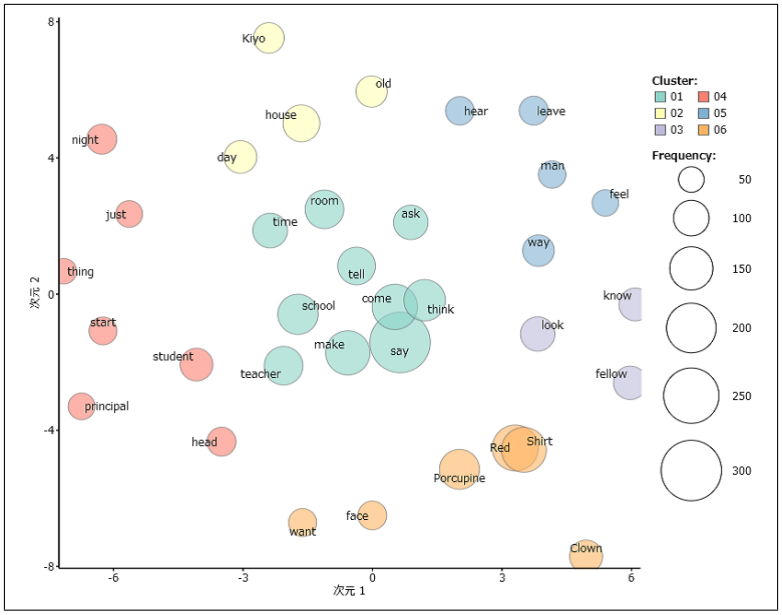

図 3.6: データから作成した多次元尺度法

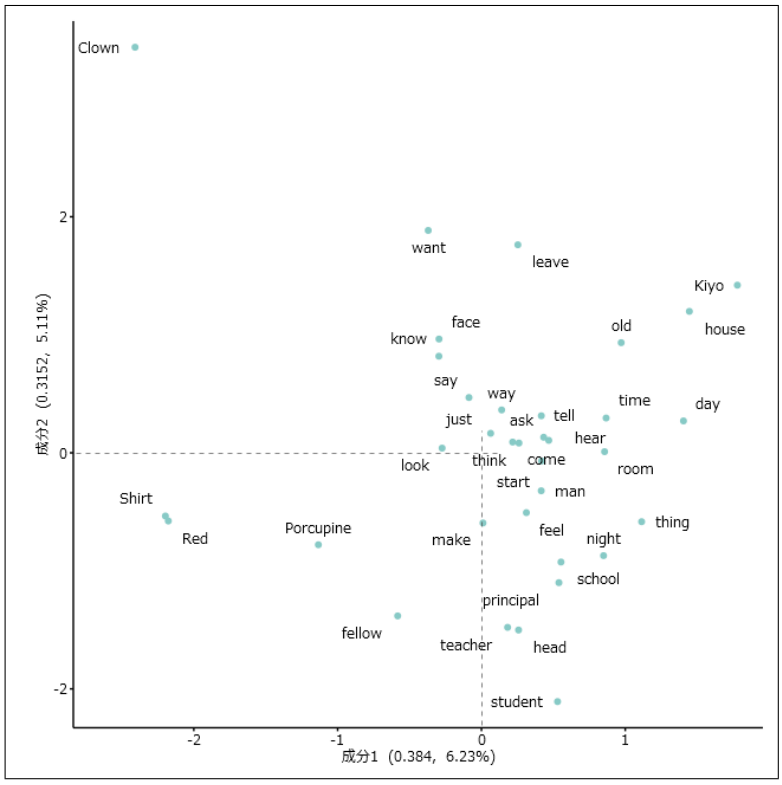

図 3.7: データから作成した対応分析

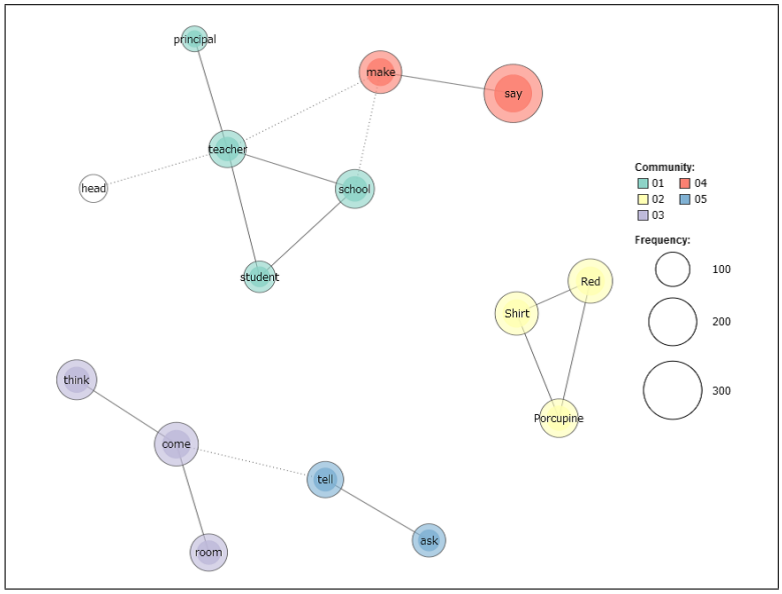

図 3.8: データから作成した共起ネットワーク

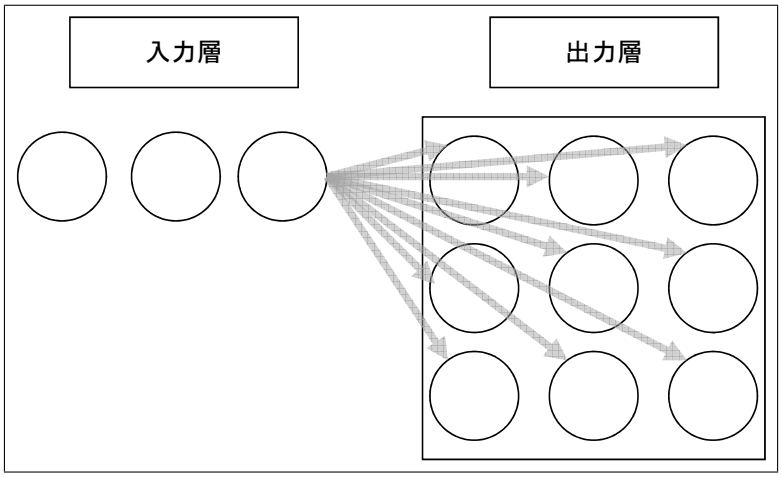

図 3.9: 入力層と出力層

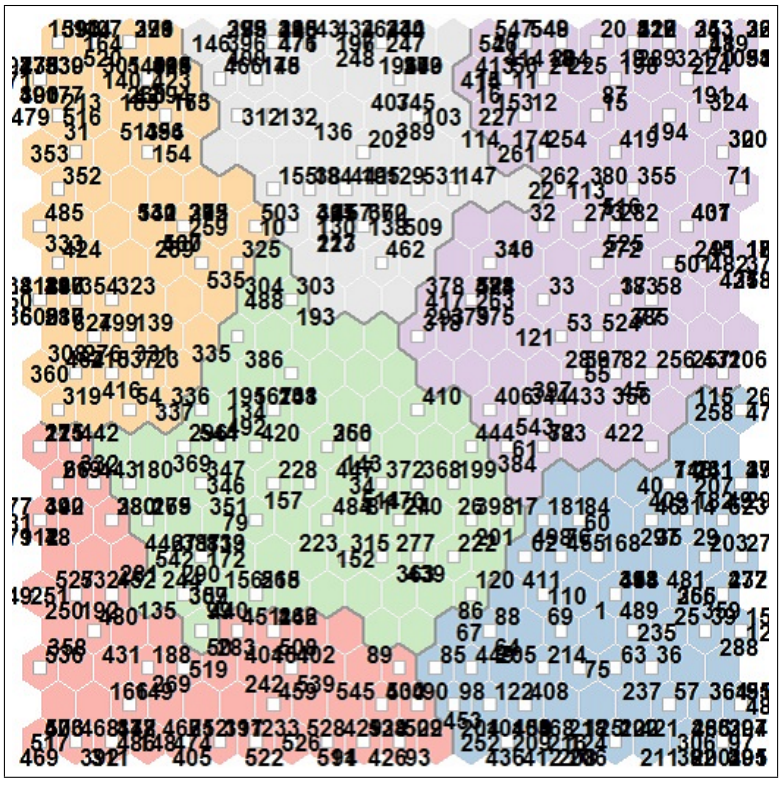

図 3.10: データから作成した SOM

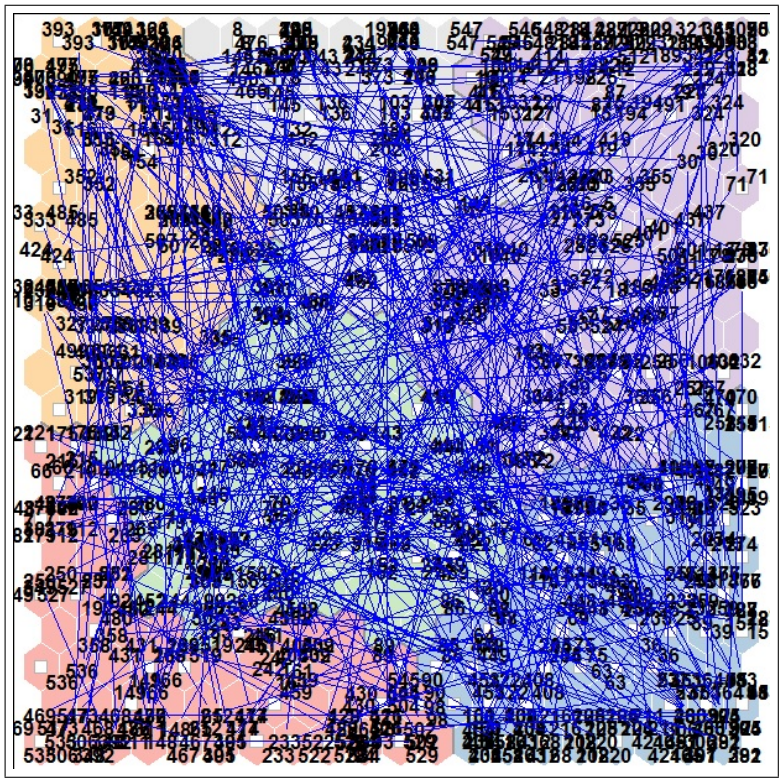

図 3.11: データから作成した SOM に id 順に繋いだ SOM

第4章

提案手法

本研究では個人情報保護に着目したライフログのため，行動識別のシステムとして，MOVERIO と画像認識 API を用いたリアルタイム視界情報テキスト変換アプリケーションの開発を行う．開発エンジンは，Unity Technologies が提供するゲーム制作向け開発エンジン Unity5 を使用する．Unity5 は 3D オブジェクトを主として扱い，モバイル端末への出力にも対応している．画像認識 API は Computer Vision API を使用し開発を行う．MOVERIO は Android5.1 であるため API level22 で Android アプリケーションを作成する．

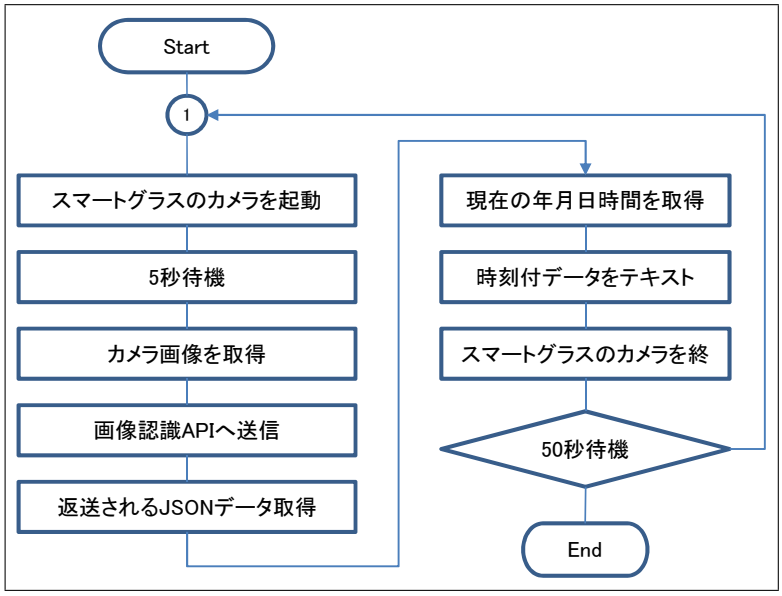

図 4.1: アプリケーションのフローチャート

図 4.1 はライフログデータ取得アプリケーションのフローチャートである．起動した際画面は真っ暗であり，カメラを起動しても画面に何も表示を行わないようにしている．MOVERIO は黒い画面は透過する性質があるため，視界を妨げずにライフログデータ取得を行える．本

表 4.1: Computer Vision API

s	tags	captions
1	indoor,black,sitting,computer,table	a black computer mouse on a table
2	indoor,black,sitting,table,computer	a black computer mouse on a table
3	indoor,black,sitting,table,computer	a black computer mouse on a table
4	indoor,black,sitting,table,mouse	a black computer mouse on a table
5	black,sitting,indoor,table,desk	a black computer mouse on a table

研究では，データが取得できているか常時確認を行うため，取得したタグを邪魔にならない程度の大きさで表示を行うプログラム（図 A.1）を使用している．

また，カメラを起動させたままだと 2 時間程度でバッテリーがなくなってしまったため，カメラの起動・終了にかかるバッテリー消耗よりも連続起動の方がバッテリー消耗が大きいと考えデータ取得後カメラを終了させている．カメラ終了をプログラムに組み込むことにより，3 時間から 3 時間半程度稼働することができた．

約一分おきにデータを取得するために，カメラの休止時間は 50 秒とし，カメラを立ち上げてから 5 秒後に撮影を行う．その後画像認識 API の応答を得るまでおおよそ 5 秒程度かかるため，約一分ごとにデータを取得できるようにしている．

表 4.1 は MOVERIO のカメラ画像を Computer Vision API に 5 回送信したときの取得結果である．この時視界にはテーブルとマウスがある状態である．Computer Vision API はタグとキャプションを取得できるが，本研究ではタグは confidence の高い順に 5 個，キャプションは confidence の最も高い一つを取得する．表 4.1 から，タグは上位 5 個にマウスが入ることは少ないが，室内を示す indoor や色が上位に表示されやすいことがわかる．キャプションの精度は高く，テーブルに置かれた黒いマウスであることを認識している．

第5章

シミュレーション結果ならびに考察

作成したアプリケーションでデータを取得した．取得日は2018年1月27日と28日の10時30分～13時30分の180分である．おおよそ一分に一回データを取得し，デバイスの処理能力に波があることからデータ数は190となっている．デバイスが充電100%である状態から充電が切れるまで取得を行ったため180分となっている．この時，抽出語の中でも，3回以上出現する抽出語を用いる．解析に使用する品詞は名詞と形容詞に絞る．理由として，動詞は取得したデータ内のキャプションに現れるが，コンピューターが置いてある，という状態をコンピューターが座っている，などのように画像認識APIが捉えるため，本研究では座っている，という状態がノイズになってしまう．そのため品詞を絞ることとした．取得したデータからKH coderでSOM，階層的クラスター分析，MDS，対応分析，共起ネットワークを行う．

27日は学校でPC作業を行い，外出（帰宅）するという流れであり，いわゆる平日の行動を表しているため平常日とする．28日は自宅でPC作業と食事を行うといういつもとは異なる行動を行っているイベント日とする．また自宅は学校よりも視界に入る物体が多い．図5.1は平常日とイベント日のタイムスケジュールである．

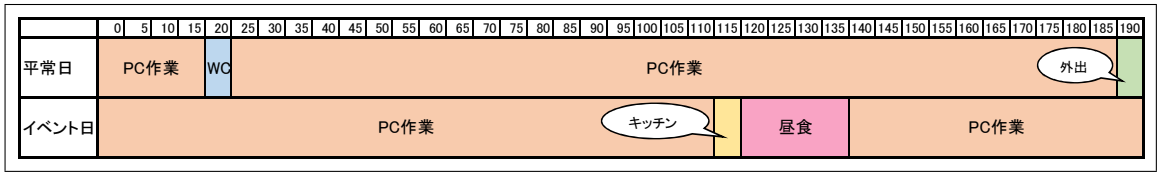

図 5.1: 平常日とイベント日のタイムスケジュール

図5.2は平常日のデータから作成したクラスター分析である．この時クラスター数はAutoでは4となり，図5.3より4前後の傾きに大きな変化がないためクラスター数は4とした．クラスター分析より，赤のクラスターはPC作業であり最も多く，青のクラスターは外出時のことを表していると考えられる．紫のクラスターは女子トイレの壁の色であるwhiteが出現していることからトイレに行くことを示しているように考えた．図5.4はイベント日

のデータから作成したクラスター分析である．この時クラスター数は Auto では 5 となり，図 5.5 より．4 よりも 5 が適切であることは明らかのためクラスター数は 5 とした．最も多いのは青のクラスターの PC 作業であることが分かり，ピンクのクラスターは食事を表し，緑のクラスターはキッチンを表していると考えるが，自室の私物も含まれてしまっている．図 5.2 と図 5.4 を比較すると，イベント日の自宅の方が物が多いことからライフログデータに関係のないものまで取得されていることがわかる．また，イベント日の方が平常日よりもクラスター数が増え，視界情報の差から併合標準もわかりやすい形となっている．

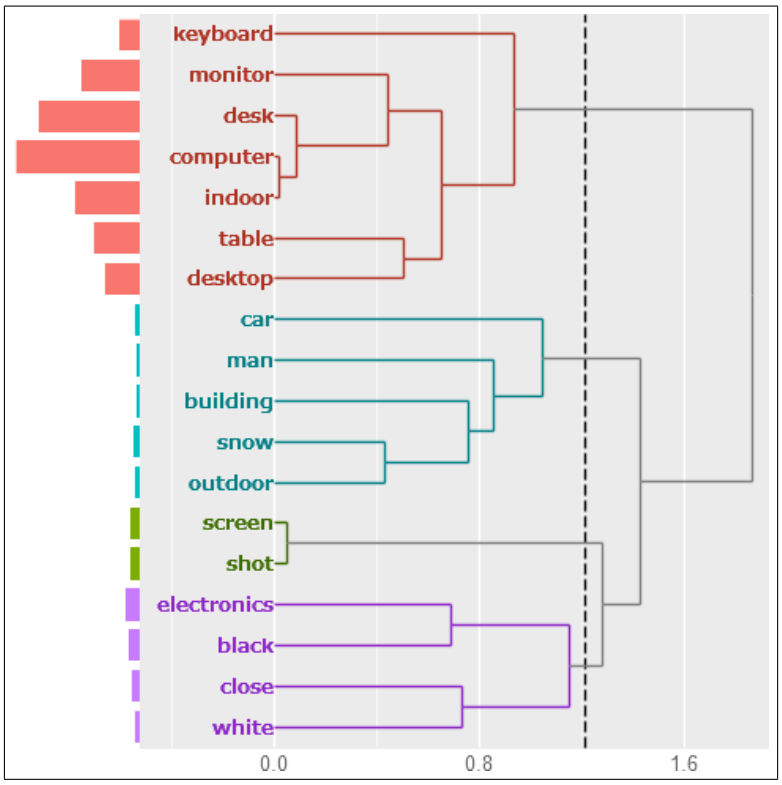

図 5.2: 平常日のデータから作成したクラスター分析

図 5.6 は平常日のデータから作成した多次元尺度法である．クラスター分析より，クラスター数は 4 とした．一番大きいクラスター 01 とクラスター 02,03,04 は距離が離れていることから行動がはっきり分かれていることがわかる．なお，クラスター 04 は 03 に近いものと考える．図 5.7 はイベント日のデータから作成した多次元尺度法である．クラスター分析より，クラスター数は 5 とした．クラスター 01 と 02 は行動として近いものであり，01 と 03 が近いのは，食事の際視界に PC が入り込んでいた影響だと考える．したがって，作業を行いながら食事を行っていることがわかる．01 から少し離れた 04 にオーブンという抽出語があるため，キッチンでオーブンを使用したことなどが考えられる．また，05 は 01〜04 より少し離れているためノイズ的であると考えられる．図 5.6 と図 5.7 を比較すると，イベント日のほうが行動を示すものが多いため全体的にクラスター間に平常日よりも大きな距離が出てこない．クラスター 01 と他のクラスターとの関係性は類似しているが，距離の密度から平常日とイベント日の相違が視覚的に判断できる．

図 5.8 は平常日のデータから作成した対応分析である．図 5.8 より，PC 作業を行ってい

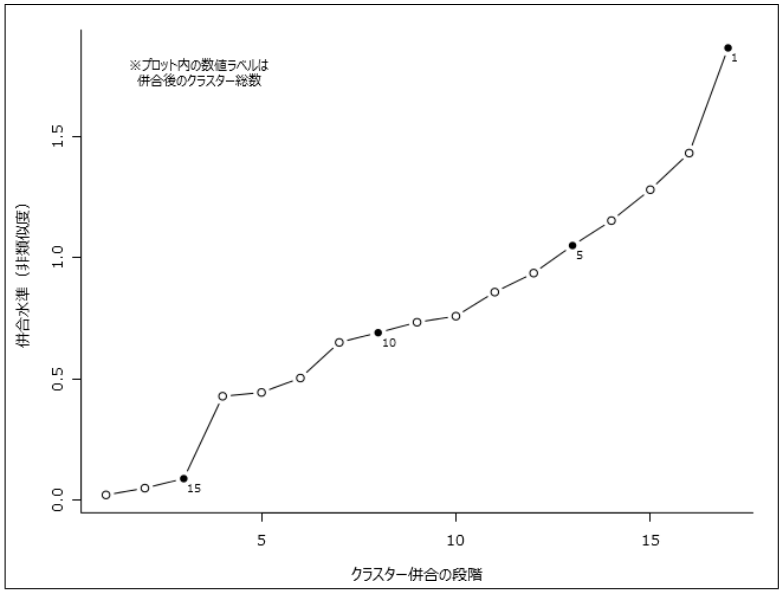

図 5.3: 平常日のクラスター分析の併合標準

ることが最も多く，特徴的でないことがわかる．また，white や outdoor は原点より離れているため特徴的な行動であると考える．図 5.9 はイベント日のデータから作成した対応分析である．図 5.8 と図 5.9 の比較より，平常日もイベント日も PC 作業を中心に行っているが，イベント日は原点より少し離れたところに oven があり，food や white はより特徴的になっていることがわかる．

図 5.10 は平常日のデータから作成した共起ネットワークである．この時，Jaccard 係数が 0.2 以上の共起関係を描画している．図 5.10 より，PC 作業を表す抽出語どうしは線で結ばれているため，共起関係があることがわかる．図 5.11 はイベント日のデータから作成した共起ネットワークであり，図 5.10 と比較すると，Jaccard 係数が 0.2 以上の強い共起関係を持つ抽出語が少なく，クラスターも少なっていることがわかる．

最後に今までの解析を踏まえて SOM を作成する．クラスター数は平常日が 4，イベント日が 5 となっている．図 5.12 から平常日は一つのクラスターを中心に行動を行っているのがわかるが，図 5.13 からイベント日は食事関係を表す黄色のクラスター以外は，常に同じ行動を行っていても，視界に入る物体の変化から線が乱雑になっている．クラスター間を大きくまたぐ線などもあり，線の形状からイベント日であることが読み取れる．

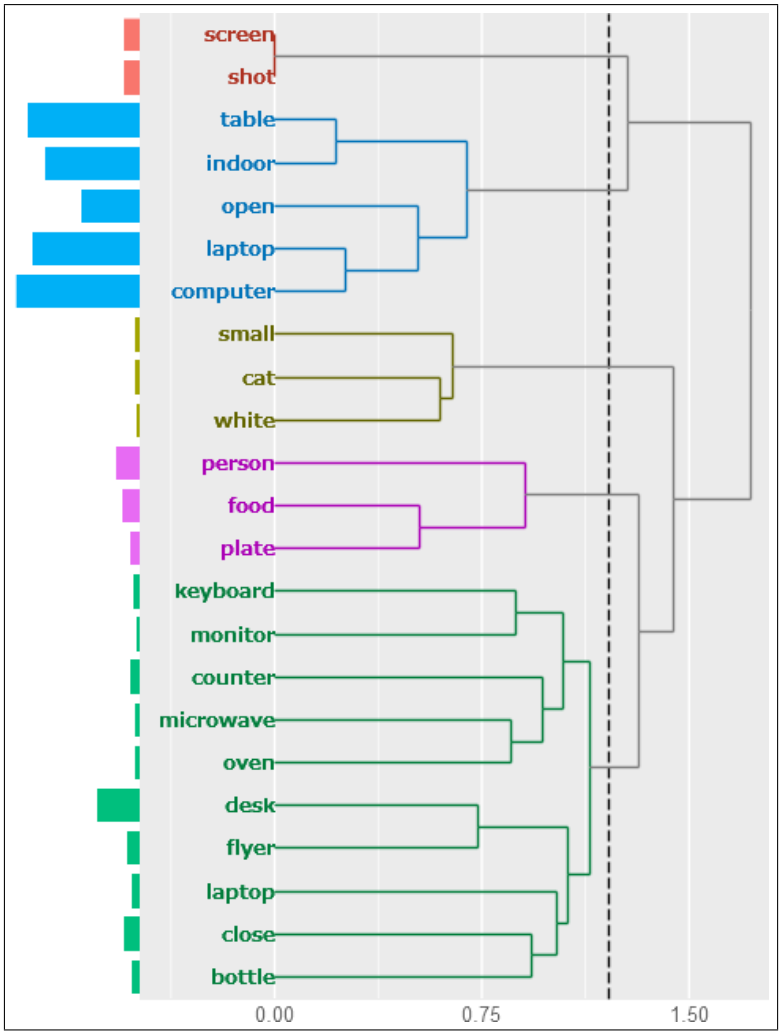

図 5.4: イベント日のデータから作成したクラスター分析

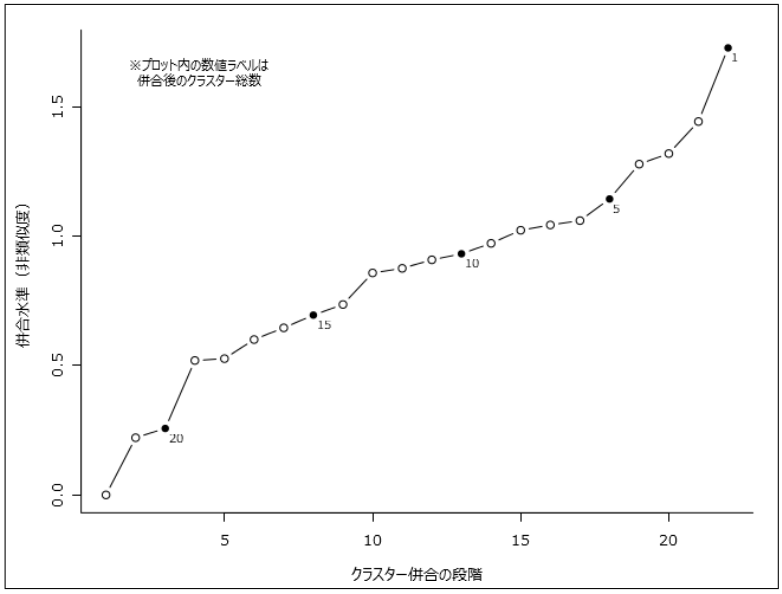

図 5.5: イベント日のクラスター分析の併合標準

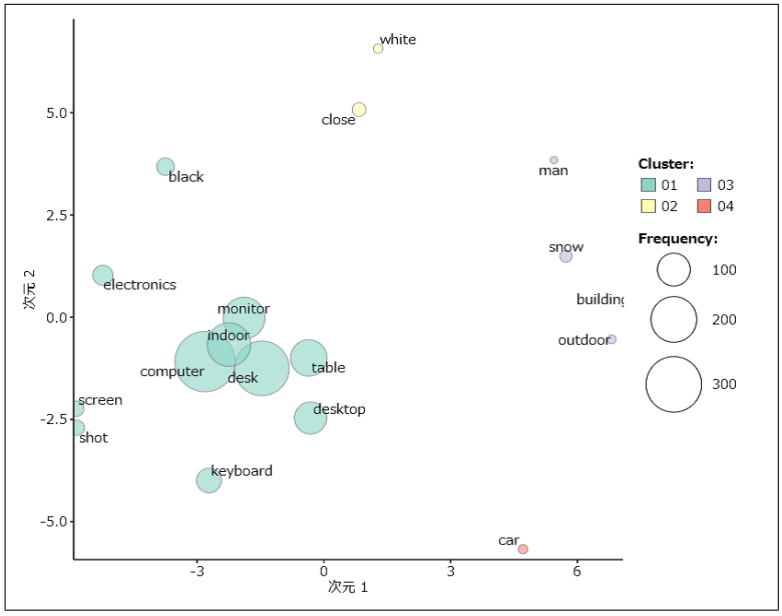

図 5.6: 平常日のデータから作成した多次元尺度法

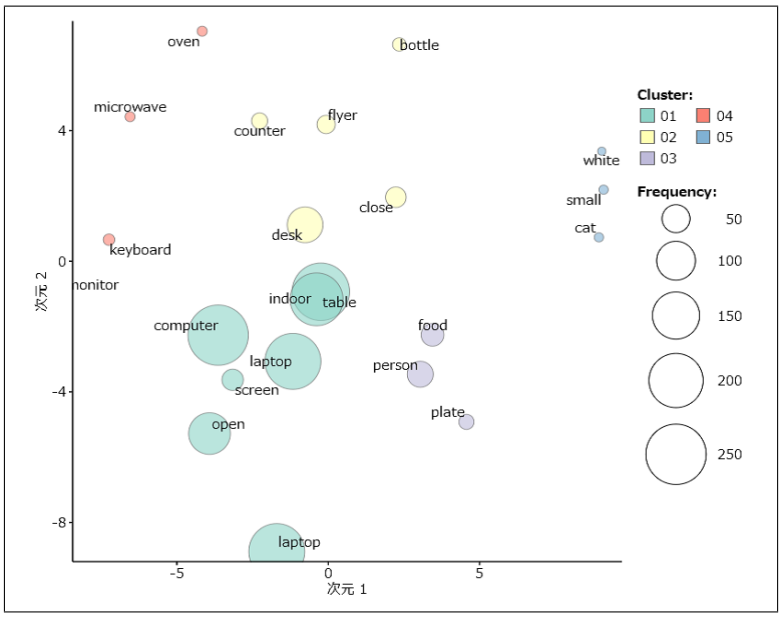

図 5.7: イベント日のデータから作成した多次元尺度法

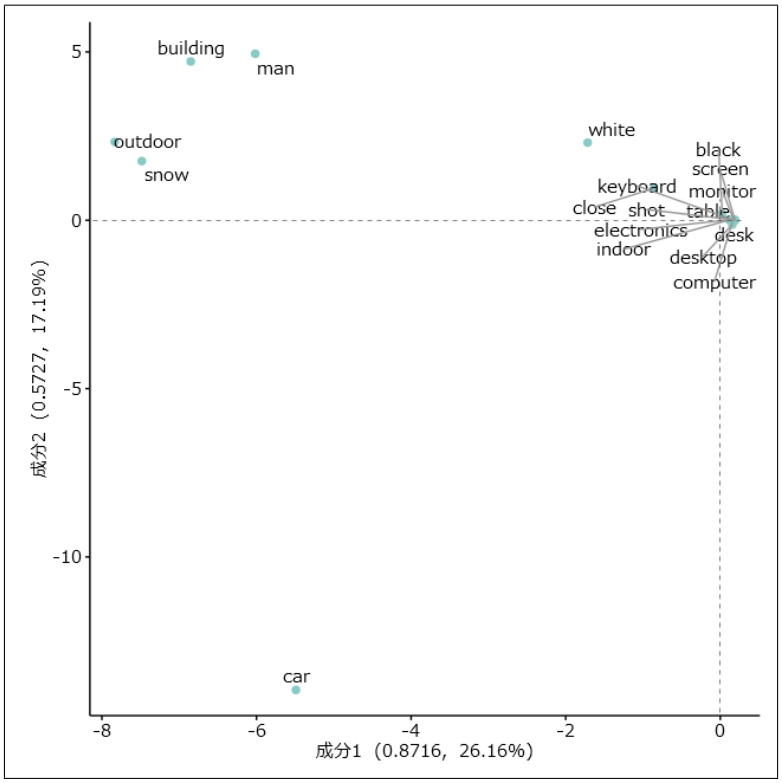

図 5.8: 平常日のデータから作成した対応分析

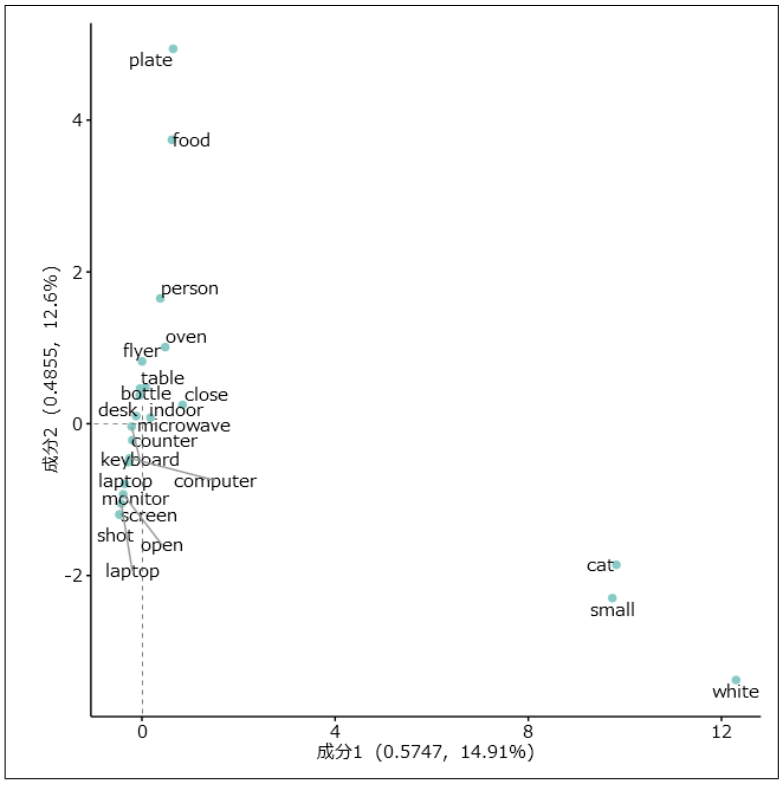

図 5.9: イベント日のデータから作成した対応分析

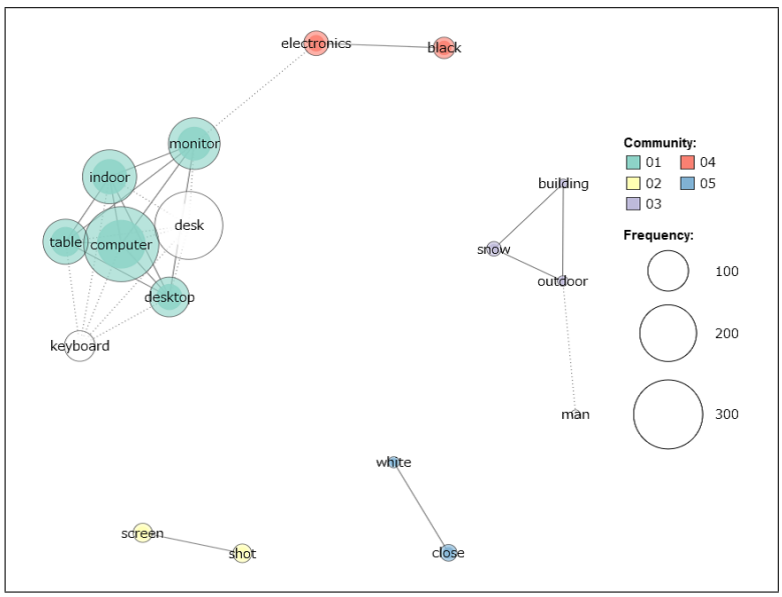

図 5.10: 平常日のデータから作成した共起ネットワーク

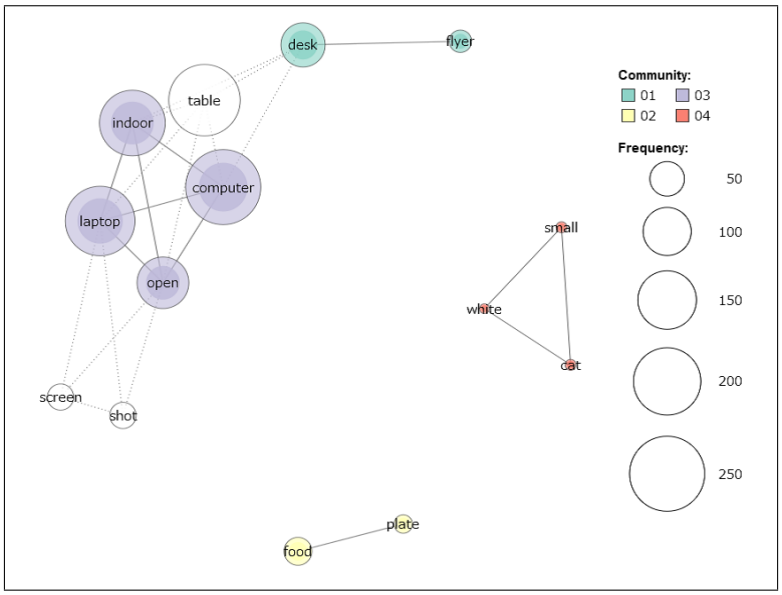

図 5.11: イベント日のデータから作成した共起ネットワーク

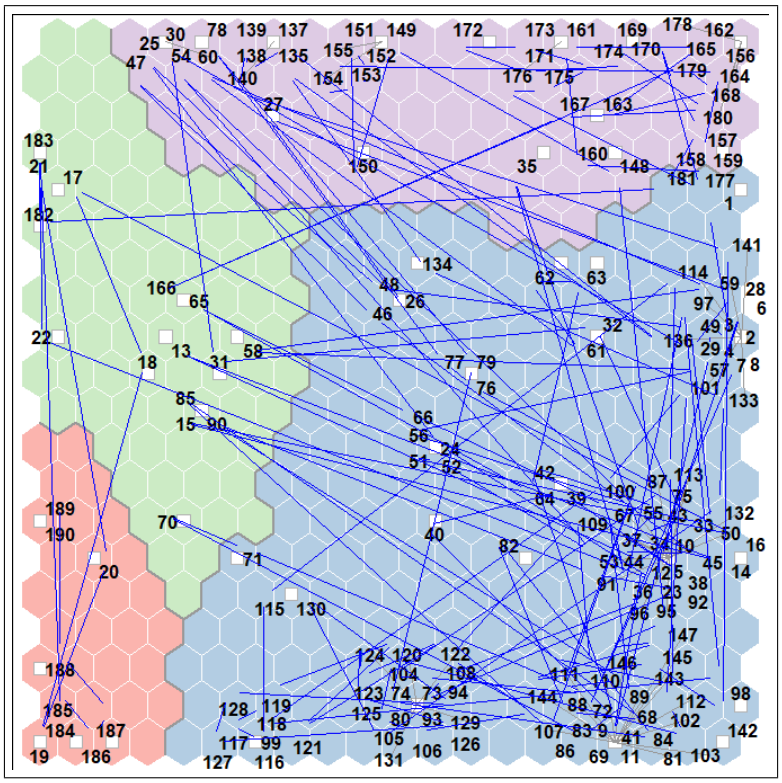

図 5.12: 平常日のデータから作成した SOM

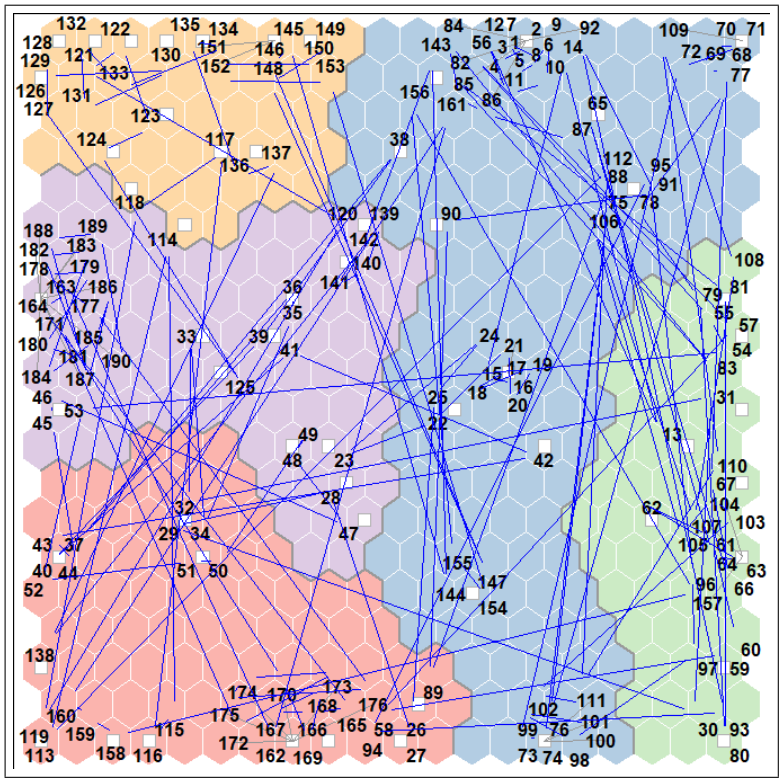

図 5.13: イベント日のデータから作成した SOM

第6章

結論と今後の展望

本研究の目的は，多くの人に広く受け入れられるライフログとして，個人情報保護に着目し，手間がかからず自動的にライフログデータの取得を行い，取得したデータから類似性やイベント性を考察できることである．開発したライフログデータ取得アプリケーションを使用したビッグデータ構築・データ解析を行い，行動パターンの類似性・イベント検出を行った．

本研究の特徴は三点あげられる．一つ目は，既存の研究では，GPS を使用することが多く，個人情報保護に着目する研究は少ないため，本研究の特徴はライフログデータ取得アプリケーションの中でも個人情報保護に着目し開発を行った点である．二つ目は，MOVERIO を使用することで，ユーザーに負担のないライフログデータ取得を行える点である．三つめは，取得したライフログデータから，多変量解析と可視化を行うことで，ライフログデータの類似性やイベント性を考察した点である．

結果として，個人情報保護に着目したライフログデータ取得アプリケーションの開発ができ，多変量解析を用いることでライフログの可視化を行い行動パターンの類似性やイベント性を視覚的に検出するという目標は達成できた．目標を達成したうえで，開発したアプリケーションの改善点を上げる．開発したアプリケーションは自動的にライフログデータを取得する点が利点として挙げられるが，一方で客観的なライフログデータしか取得できないという弱点もある．ユーザーが興味を持った瞬間や，データを取得したい瞬間のライフログデータは現状のアプリケーションには含まれていないためである．この弱点に対し，ユーザーが取得したいタイミングでライフログデータを取得する方法をアプリケーションに組み込む必要がある．組み込むため，取得したいタイミングを MOVERIO に伝える方法の検討も必要となる．今後の展望として，ユーザーの取得タイミングを組み込んだライフログデータ取得アプリケーションの開発が挙げられる．また，長時間のライフログデータ取得にデバイスが動作に耐えうるようプログラムの改善を行うことが求められる．

本研究の研究成果は，テキストによるライフログデータ取得，解析を行い新たなビジネスプランの検討やユーザー自身の生活の見直しなどに使用できるため，より高度なアプリケーション開発を目指す開発者，研究者の方々の参考になれば幸いである．解明できた点は必ずしも多くはないが，若干なりとも寄与できたと思われる．

謝辞

本研究を進めるにあたり，ご指導を頂きました奥原浩之教授に感謝致します．また，多くの知識や示唆を下さいました奥原研究室の同期・後輩の皆様へ心から感謝の気持ちと御礼を申し上げたく，謝辞にかえさせていただきます．

2018 年 2 月

福嶋 瑞希

参考文献

[1] 緒方広明.　日本語学習を支援するユビキタス学習環境に関する研究.　http://www.taf.or.jp/files/items/542/File/P212.pdf. 閲覧日 2018,1,30.

[2] 宏貴角田, Hiroki SUMIDA. ライフログ分析による行動特徴抽出及びイベント検出. 法政大学大学院紀要（情報科学研究科編）, Vol. 9, pp. 119–124, mar 2014.

[3] 裕司矢野, 健横井, 智訓橋山. Ro-003 行動辞書を利用した twitter からの行動抽出 (分析とモデリング,o 分野:情報システム). 情報科学技術フォーラム講演論文集, Vol. 11, No. 4, pp. 51–56, sep 2012.

[4] 新保史生. ライフログの定義と法的責任　個人の行動履歴を営利目的で利用することの妥当性. 情報管理, Vol. 53, No. 6, pp. 295–310, 2010.

[5] 「ナニコレ！ヨコハマ？写真＆ウォーキング」開催しました　——wordpress × junaio の応用事例：反省と考察. http://takelab.sub.jp/2012/11/26/ 閲覧日 2018,1,30.

[6] 芳竹宣裕, 伊藤慎. ユビキタス環境が生み出す大量情報 「ライフログ」 の活用と実装技術. NEC 技報, Vol. 62, No. 4, p. 77, 2009.

[7] 川上晃平. スマートグラスを利用した授業支援システムの開発. 2017.

[8] 倉田陽平・真田 風・鈴木祥平・石川博. Flickr と google cloud vision api によりテーマ別観光マップを作る試み. http://db-event. jpn.org/deim2017/papers/321.pdf, 2017. 閲覧日 2018,1,4.

[9] 雄治, 吉川眞, 田中一成. ソーシャルメディアを活用した景観の分析と評価. 日本都市計画学会関西支部研究発表会講演概要集, Vol. 15, pp. 13–16, 2017.

[10] 亜令小林, 健嗣岩本, 智西山. 釈迦：携帯電話を用いたユーザ移動状態推定・共有方式 (モバイルコンピューティング, モバイルアプリケーション, ユビキタス通信, モバイルマルチメディア通信). 電子情報通信学会技術研究報告. MoMuC, モバイルマルチメディア通信, Vol. 108, No. 44, pp. 115–120, may 2008.

[11] 努寺田. ウェアラブルセンサを用いた行動認識技術の現状と課題. コンピュータ ソフトウェア, Vol. 28, No. 2, pp. 43–54, 2011.

[12] 貴志一樹, 山崎俊彦, 相澤清晴. 7-5 机上行動のライフログのための行動認識 (第 7 部門 ヒューマンインタフェース). 映像情報メディア学会年次大会講演予稿集, Vol. 2014, , 2014.

[13] 卓也前川, 豊柳沢, 泰恵岸野, 勝彦石黒, 剛次亀井, 保志櫻井, 剛岡留. ウェアラブルセンサによるモノを用いた行動の認識について. Technical Report 57, NTT コミュニケーション科学基礎研究所, NTT 西日本, NTT コミュニケーション科学基礎研究所, NTT コミュニケーション科学基礎研究所, 国際電気通信基礎技術研究所, NTT コミュニケーション科学基礎研究所, 関西学院大学, mar 2010.

[14] 耕一樋口. テキスト型データの計量的分析：2つのアプローチの峻別と統合. 理論と方法, Vol. 19, No. 1, pp. 101–115, mar 2004.

[15] 香織佐野, 在鎬李, かおりさの, じぇほり. Kh coderで何ができるか：日本語習得・日本語教育研究利用への示唆. 言語文化と日本語教育, Vol. 33, pp. 94–95, jun 2007.

[16] Kh coderを用いた研究事例のリスト. http://khc.sourceforge.net/bib.html, 2017. 閲覧日 2018,1,7.

[17] 二宮隆次, 小野浩幸, 高橋幸司, 野田博行. 新聞記事を基にしたテキストマイニング手法による産学官連携活動分析. 科学・技術研究, Vol. 5, No. 1, pp. 93–104, 2016.

[18] クラスター分析の手法（階層クラスター分析） — データ分析基礎知識. https://www.albert2005.co.jp/knowledge/data_mining/cluster/hierarchical_clustering. 閲覧日 2018,1,24.

[19] 階層的クラスター分析について. http://koichi.nihon.to/cgi-bin/bbs_khn/khcf.cgi?list=&no=977&mode=allread&page=0. 閲覧日 2018,1,24.

[20] 美龍李, 恒也田中, 吉弘成田. 画像を用いた製品の「飽き」に関する感性評価:－デザインの視覚的要素を中心に－. 日本感性工学会論文誌, Vol. 11, No. 3, pp. 407–417, 2012.

[21] 小峯敦・下平裕之. ベヴァリッジ『自由社会における完全雇用』のケインズ的要素～テキストマイニングを加味した量的・質的分析～. 2017.

[22] 齋藤堯幸. 多次元尺度構成法. 計測と制御, Vol. 22, No. 1, pp. 126–131, 1983.

[23] Joseph B Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, Vol. 29, No. 1, pp. 1–27, 1964.

[24] 慶一郎中山. ¡研究ノート¿対応分析によるデータ解析. 関西学院大学社会学部紀要, No. 108, pp. 133–145, oct 2009.

[25] 京子田中. Kh coderとrを用いたネットワーク分析. 久留米大学コンピュータジャーナル, Vol. 28, pp. 37–52, mar 2014.

[26] 共起ネットワークにおける中心性の解釈について. http://www.koichi.nihon.to/cgi-bin/bbs_khn/khcf.cgi?no=2493&mode=allread#2496. 閲覧日 2018,2,2.

[27] 横田尚己, 山田圭二郎. 熊本地震のつぶやきに見る感情極性値の時空間解析. 都市計画論文集, Vol. 52, No. 3, pp. 1081–1087, 2017.

[28] 正増田, Masuda Tadashi, 高崎経済大学地域政策学部. 地方議会の会議録に関するテキストマイニング分析：高崎市議会を事例として. 地域政策研究 = Studies of regional policy, Vol. 15, No. 1, pp. 17–31, aug 2012.

[29] T. KOHONEN. Self-organized formation of topologically correct feature map. *Biol. Cybern.*, Vol. 43, pp. 59–69, 1982.

[30] 自己組織化マップを用いた気象要素の分類と予測. http://www.gifu-nct.ac.jp/elec/deguchi/sotsuron/oka/oka.html, 2011. 閲覧日 2018,1,7.

[31] 自己組織化特徴マップ（som）. http://www.sist.ac.jp/ kanakubo/research/neuro/selforganizingmap.html. 閲覧日 2018,1,31.

付録

A. 1　ライフログデータ取得アプリケーションのソースコード

ライフログデータ取得アプリケーションのソースコードを図 A.1 にしめす.

図 A. 1: som.cs

```
1  using UnityEngine;
2  using System.Collections;
3  using System.Collections.Generic;
4  using UnityEngine.UI;
5  using System.IO;
6  using System;
7  using System.Text;
8  using System.Linq;
9
10 public class SumDataFromJson
11 {
12         public SumDataFromJsonDescription description;
13 }
14
15 [System.Serializable]
16 public class SumDataFromJsonDescription
17 {
18         public string[] tags;
19 }
20
21 public class som : MonoBehaviour
22 {
23         private float captureIntervalSeconds = 50.0f;
24         private float captureIntervalSeconds2 = 5.0f;
25         public Text gtext;
26         Dictionary<string, string> headers;
27         private int Width = 1280;
28         private int Height = 720;
29         private int FPS = 30;
30         private WebCamTexture webcamTexture;
31         private Color32[] color32;
32         private string tagsStg;
33         private string reportFileName = "cut_report.txt";
34         private string reportFileName2 = "long_report.txt";
35         public bool addDateTime = true;
36
37         void Start ()
38         {
39                 Screen.sleepTimeout = SleepTimeout.NeverSleep;
40                 StartCoroutine ("Sample");
41         }
42
43         public IEnumerator Sample ()
44         {
45                 WebCamDevice[] devices = WebCamTexture.devices;
46                 WebCamDevice userCameraDevice = WebCamTexture.devices [0];
```

```
47                  webcamTexture = new WebCamTexture (userCameraDevice.name,
                        Width, Height, FPS);
48                  webcamTexture.Play ();
49                  Debug.Log ("webcamTexture");
50
51                  yield return new WaitForSeconds (captureIntervalSeconds2);
52                  color32 = webcamTexture.GetPixels32 ();
53                  Texture2D texture = new Texture2D (webcamTexture.width,
                        webcamTexture.height);
54                  texture.SetPixels32 (color32);
55                  texture.Apply ();
56                  byte[] jpg = texture.EncodeToJPG ();
57                  string VISIONKEY = "ba7982e18b4943d18024749aca8031fb";
58                  var uri = "https://westus.api.cognitive.microsoft.com/vision/v1
                        .0/describe";
59
60                  string responseData;
61                  var headers = new Dictionary<string, string> () {
62                          { "Ocp-Apim-Subscription-Key", VISIONKEY },
63                          { "Content-Type", "application/octet-stream" }
64                  };
65
66                  WWW www = new WWW (uri, jpg, headers);
67                  yield return www;
68                  responseData = www.text;
69                  Debug.Log (responseData);
70
71                  var data = JsonUtility.FromJson<SumDataFromJson> (responseData);
72                  if (data != null && data.description.tags != null) {
73                          tagsStg = data.description.tags.Take (5).Aggregate ((a, b) => a
                                + ",␣" + b);
74                          Debug.Log (tagsStg);
75                          gtext.text = tagsStg;
76
77                          DateTime dt = DateTime.Now;
78                          string text = dt.ToString ("[yyyy-MM-dd␣HH:mm:ss]") +
                                tagsStg.ToString () + "\n";
79                          string text2 = dt.ToString ("[yyyy-MM-dd␣HH:mm:ss]") +
                                responseData.ToString () + "\n";
80
81                          string outfile = reportFileName;
82                          string outfile2 = reportFileName2;
83
84                          if (addDateTime) {
85                                  string file = Path.GetFileNameWithoutExtension (
                                        reportFileName);
86                                  string ext = Path.GetExtension (reportFileName); //を
                                        含む拡張子”.”
87                                  outfile = file + "_" + dt.ToString ("yyyyMMdd") + ext;
88
89                                  string file2 = Path.GetFileNameWithoutExtension (
                                        reportFileName2);
90                                  string ext2 = Path.GetExtension (reportFileName2); //
                                        を含む拡張子
                                        ”.”
91                                  outfile2 = file2 + "_" + dt.ToString ("yyyyMMdd") +
                                        ext2;
92                          }
93
94                          SaveText (text, Path.Combine (Application.persistentDataPath,
```

```
                    outfile));
95                  SaveText (text2, Path.Combine (Application.persistentDataPath,
                        outfile2));
96                  MonoBehaviour.Destroy (texture);
97                  MonoBehaviour.Destroy (webcamTexture);
98                  color32 = null;
99          }
100         StartCoroutine ("StopRunTimeTemp");
101     }
102
103     public IEnumerator StopRunTimeTemp ()
104     {
105         webcamTexture.Stop ();
106         gtext.text = tagsStg;
107         yield return new WaitForSeconds (captureIntervalSeconds);
108         StartCoroutine ("Sample");
109     }
110
111     public static bool SaveText (string text, string path)
112     {
113         try {
114             using (StreamWriter writer = new StreamWriter (path, true))
                    {
115                     writer.Write (text);
116                     writer.Flush ();
117                     writer.Close ();
118                 }
119         } catch (Exception e) {
120             Debug.Log (e.Message);
121             return false;
122         }
123         return true;
124     }
125 }
```