

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

Similarity and event detection by behavior pattern analysis from
environment recognition life log

富山県立大学 電子・情報工学科

1415048 福嶋 瑞希

指導教員 奥原 浩之 教授

平成30年2月6日

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	X, Y
クラスター内での重心とサンプルとの距離の 2 乗和	$L(X), L(Y)$
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	$L(X \cup Y)$
入力データベクトル	x
競合層のニューロンの番号	i
参照ベクトル	m_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
学習回数	t
学習率係数	$\alpha(t)$
学習半径	$\sigma^2(t)$

目次

記号一覧	1
第1章 序論	5
§ 1.1 本研究の概要	5
§ 1.2 本研究の目的	5
§ 1.3 本論文のあらまし	6
第2章 ライフログとスマートグラス	7
§ 2.1 現状のライフログ	7
§ 2.2 スマートグラス	8
§ 2.3 画像認識 API	9
第3章 行動識別	11
§ 3.1 行動識別	11
§ 3.2 行動識別のための分析手法	12
§ 3.3 類似性・イベント性	17
第4章 提案手法	21
§ 4.1 開発システムの概要	21
§ 4.2 省電力化	22
§ 4.3 周期性の検出	23
第5章 シミュレーション結果ならびに考察	25
第6章 結論と今後の課題	33
謝辞	34
参考文献	35
付録	38
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	38
A. 2 時系列 SOM を作成するソースコード	40

序論

§ 1.1 本研究の概要

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ライフログデータは、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 [1]。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い [2] [3]。また、手動でライフログデータを取得する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する。

§ 1.2 本研究の目的

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取得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（Self-organizing maps：SOM）、階層的クラスター分析、多次元尺度構成法（Multi Dimensional Scaling：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。

§ 1.3 本論文のあらまし

本論文は次のように構成される。

第1章：本章 第一章では、本研究の概要と目的について説明した。

第2章 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

第3章 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 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 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。

第6章 まとめと今後の課題を述べる。

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「Lifelog¹」がある (図 2.1 参照)。このアプリケーションはスマートウェア「SmartBand 2²」 (図 2.2 参照) 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³」 (図 2.3 参照) や、手動でマッピングできる「Swarm⁴」 (図 2.4 参照) 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 (Global Positioning System, Global Positioning Satellite : 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 [4] 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 [5] [6]。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 [7]。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

¹<http://www.sonymobile.co.jp/myxperia/app/lifelog/>

²<http://www.sonymobile.co.jp/product/smartproducts/swr12/>

³<https://play.google.com/store/apps/details?id=org.liteapp.mat2>

⁴<https://play.google.com/store/apps/details?id=com.foursquare.robin>

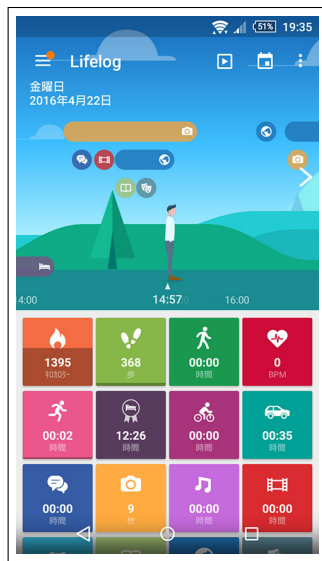


図 2.1: Lifelog



図 2.2: SmartBand 2

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 [2] [3]. 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、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 Edition⁵ (図 2.5 参照), SmartEyeglass⁶ (図 2.6 参照) 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 [8] ため、両手を常にか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 [9]。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。

このように、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。この利点を生かし、ユーザーに負担をかけることなくライフログデータを取得できる。

⁵<https://www.x.company/glass/>

⁶<https://developer.sony.com/ja/develop/smarteyeglass-sed-e1/>

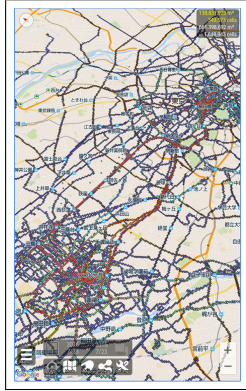


図 2.3: マッピング-GPS ログまとめて全部記録

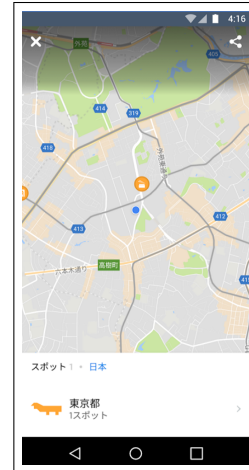


図 2.4: Swarm



図 2.5: Glass Enterprise Edition



図 2.6: SmartEyeglass

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 2.7)。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を使用することが可能である [8]。ビデオシースルー方式は、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方式である。一方、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。

§ 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。画像認識技術を用いることで、画像が持つ情報をテキストで取得することができる。代表的な画像認識 API に、Google Cloud Vision API と、Computer



図 2.7: MOVERIO BT-300

Vision API, IBM Bluemix Alchemy API という三種類がある。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 API⁷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 [10]、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 API⁸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 [11]。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。

IBM Bluemix Alchemy API

Watson が提供している API の一つである IBM Bluemix Alchemy API⁹は、画像に対してタグ付けを行うことができる。キャプションは出力できないが、分類結果の階層構造に強く、食べ物の分類などに特化している。Lite コースは無料である

⁷<https://cloud.google.com/vision/>

⁸<https://azure.microsoft.com/ja-jp/services/cognitive-services/computer-vision/>

⁹<https://www.ibm.com/watson/jp-ja/developercloud/visual-recognition.html>

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ライフログデータとして取得し、取得したライフログデータの解析から行動を認識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識別を行う研究 [12] 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取得し、行動識別を行う [13] 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ライフログデータとして取得することは難しい。机上に設置した KinectTM(図 3.1 参照)を用いて机上の細かい動作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行動は認識できない [14]。なお、KinectTM は 2017 年 10 月 25 日に生産終了が公表されている。

本研究ではGPSやKinectTMを使用せず、細かい行動をライフログデータとして取得する手法として、ユーザーの視界情報を取得する。視界情報を取得することで、視界上にある物体からどのような作業を行っているのか、視界の風景から室内か室外にいることなどを判断できるため細かい行動をライフログデータとして取得することが可能となる。しかし、ユーザーの視界情報を取得すると、GPSを使用しなくても、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してしまう。この結果、ユーザーやユーザーの視界にいる第三者のプライバシーを侵害してしまう。さらに、視界情報を画像や動画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 [15]。本研究では MOVERIO のカメラ画像を取得し、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ライフログデータの取得を検討する。



図 3.1: Kinect™

§ 3.2 行動識別のための分析手法

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し、行動識別を行う。そのために、多変量解析である 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用いてテキストデータの可視化を行う。

多変量解析を行うツールとして KH Coder⁸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 [16]。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、SOM、共起ネットワークなどの多変量解析を行ったりできる [17]。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 [18]。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 参照）。この「坊ちゃん」英語版テキストデータは KH Coder をダウンロードする際に同時に取得することが可能であり、KH Coder のホームページよりダウンロード⁹後 C:\khcoder3\tutorial.en にテキストファイルとして保存されている。

KH Coder をダウンロードする際に取得できる KH Coder3 リファレンス・マニュアルによると、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）を読み込むことから始める。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ことで、分析対象から外すことができる（図 3.3 参照）。この Stop words 一覧テキストファイルは、KH Coder をダウンロードする際に同時に取得することが可能であり、解析を行う際に必要でない単語を自由に追加できる。

次に、前処理として、POS Tagger¹⁰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 [19] (表 3.2 参照) 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

⁸<http://khc.sourceforge.net/>

⁹<http://khc.sourceforge.net/dl3.html>

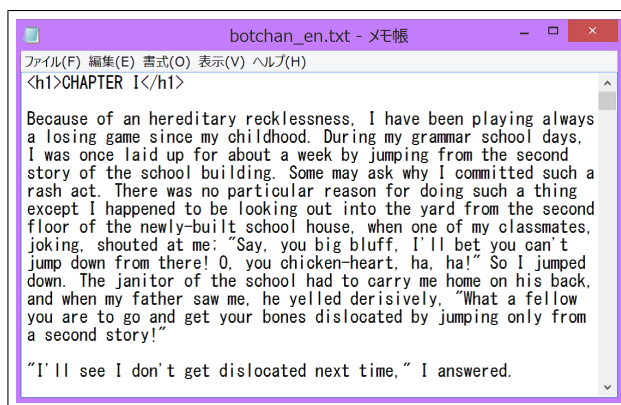


図 3.2: 「坊ちゃん」英語版テキストデータの一部

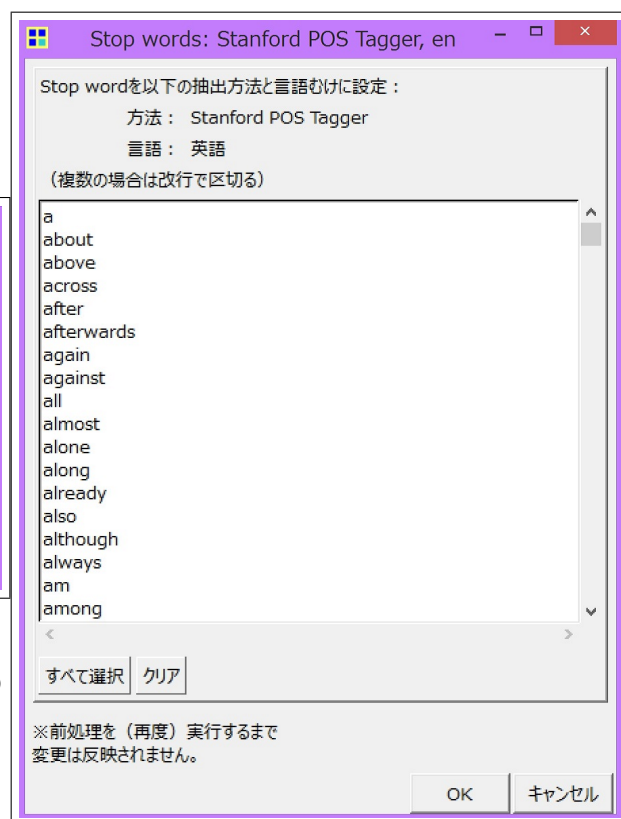


図 3.3: Stop words の一部

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.

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, MDS, 対応分析, 共起ネットワークを行う. 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 [20]. 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を用いている [21]. 1つのクラスターには関連性が高い抽出語が集まっているため, クラスターごとに集まっている抽出語を調べることでテキストデータ全体における文書の傾向や特徴を知ることができる. 階層的クラスター分析の作成方法は, まず抽出語として A,B,C,D 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を A と B とし, A と B をくくり, 2 点の代表点を求める. 次に, AB の重心, C, D の 3 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 [20] [22].

KH Coder3 リファレンス・マニュアルによると, クラスター間の距離測定方法として,

¹⁰<https://nlp.stanford.edu/software/tagger.html>

表 3.1: 「坊ちゃん」データを用いて KH Coder で出力した抽出語の一部

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: 「坊ちゃん」データを用いて KH Coder で出力した「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

KH Coder ではワード法を使用している．2つのクラスター X, Y 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 $L(X \cup Y)$ 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 $L(X)$ ， $L(Y)$ の差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．ワード法は，計算量が多いが分類感度がいいため用いられることが多く，ワ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 [23] [24]．

$$\Delta = L(X \cup Y) - L(X) - L(Y) \quad (3.1)$$

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 参照）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

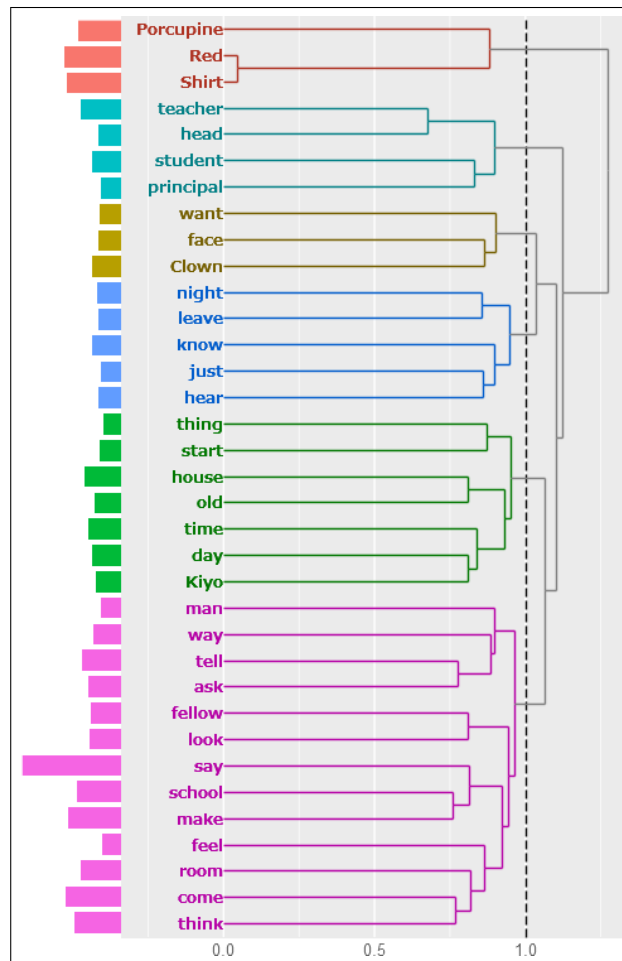


図 3.4: 「坊ちゃん」データから作成したクラスター分析

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．

MDS 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 [25]．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 MDS を使用している [26]．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，「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 [27]．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．

$$\text{Jaccard 係数} = \frac{\text{語 A と語 B を含む文書}}{\text{語 A もしくは語 B を含む文書}} \quad (3.2)$$

図 3.7 は坊ちゃんデータから出力した MDS である．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

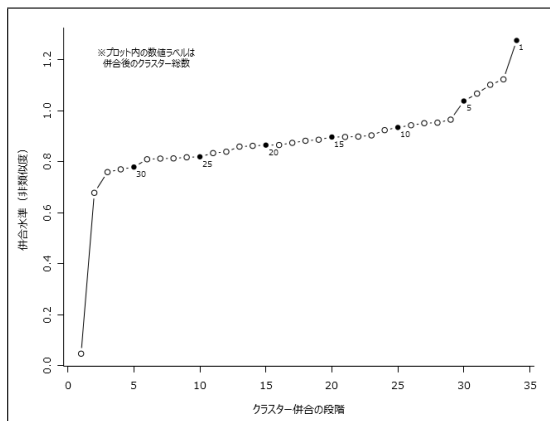


図 3.5: クラスター分析の併合標準

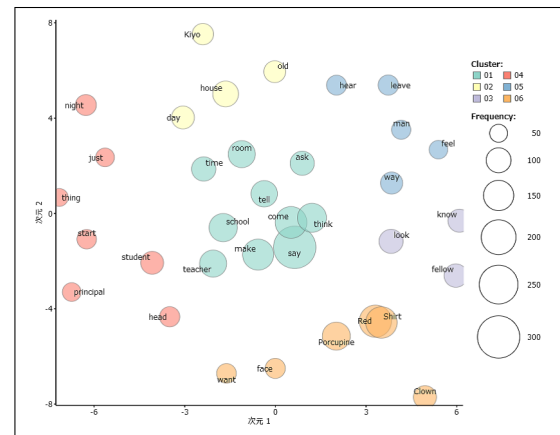


図 3.6: 「坊ちゃん」データから作成した MDS

低いクラスター・抽出語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 [28]。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。

図 3.8 は坊ちゃんデータより出力した対応分析である。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解析手法であり、線がつながっている語が共起関係にあり、その繋がりにも着目する [29]。抽出語の中で出現パターンの似通ったものを線で結ぶネットワークであり、すなわち共起関係を線で表したネットワークである。MDS 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 Coder3 リファレンス・マニュアルによると、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。種類は、中心性（媒介）、中心性（次数）、中心性（固有ベクトル）、サブグラフ検出（媒介）、サブグラフ検出（random walks）、サブグラフ検出（modularity）の 6 種類がある。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 [30]。サブグラフ検出とは、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方法 [31] であり、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語の集まりである。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 [32]。

坊ちゃんデータを使用して出力した共起ネットワークが図 3.9 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

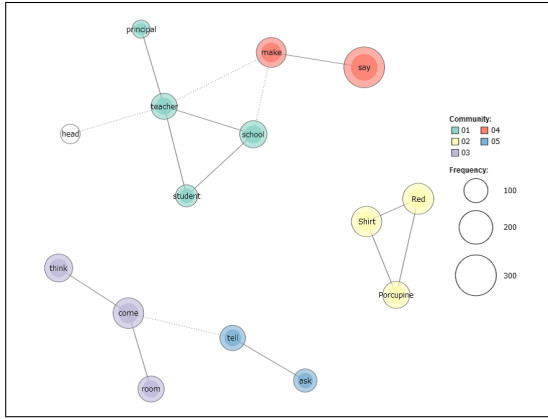


図 3.9: 「坊ちゃん」データから作成した共起ネットワーク

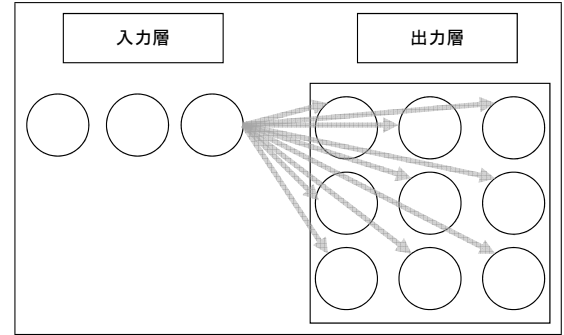


図 3.10: 入力層と出力層

次に、勝者ニューロンと勝者ニューロンに近いニューロンは自らの参照ベクトルと入力データベクトルを近づける学習を行うため、参照ベクトルを同様に更新させる。この時、 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る。

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

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

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、KH Coder3 リファレンス・マニュアルによると、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している。

KH Coder3 リファレンス・マニュアルによると、KH Coder の SOM 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」に設定されており、全体のノード数が 40 の場合は 200,000 回である。また、各ノードがもつベクトルをワ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ため、クラスター分析などの結果からクラスター数を調整する。

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。KH Coder で出力できる SOM (図 3.11 参照) は抽出語どうしの関係を示すものとなっているため、今回のライフログデータの解析に用いることには向かないためである。

KH Coder で出力できる SOM の R ファイルを基にソースコードを書き換え、R で出力を行う。出力した SOM が図 3.12 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 3.12 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

やイベント性のある行動がわかる。

SOMはクラスターの分かれ方や時系列を表すプロット順を追っていくことで行動パターンを識別し、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。

図3.13はライフログデータとして、データA, データB, データCがあり、行動識別により各データに三つの行動が存在していることが検出できた場合を表している。この時、データAとデータBは類似性があるといえる。一方でデータA, データBと、データCは類似性がないといえる。この三つのデータが一人のユーザーのライフログデータであれば、平常日とイベント日の比較に利用できる。もし、三つのデータがバラバラのユーザーである場合、ライフログの類似性があるユーザーどうしでコミュニケーションを促進することができたり。ライフログの類似性がないユーザーどうしの比較を行うことで行動の中で改善すべき行動を検出できたりする [36]。このようにライフログデータの類似性やイベント性の検出は様々な応用が可能であると考えられる。

提案手法

§ 4.1 開発システムの概要

本研究で開発するシステムは，アプリケーションを用いたデータ取得部と，多変量解析によるデータ解析を用いた行動識別部で構成される．図 4.1 はシステムの全体図である．まず，データ取得部について提案をする．

本研究では個人情報保護に着目したライフログのため，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 4.2 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（ソースコード A.1 参照）を使用している．カメラ画像を取得すると，画像認識 API へ送信する．画像認識 API を通じて，カメラ画像の情報が JSON データで取得できる．この JSON データに取得時の年月日と時刻を追加して，テキストデータとして MOVERIO 内に保存される．テキストファイルは「long_report_yyyymmdd.txt」という名前で保存され，MOVERIO 内に同じ名前のテキストファイルがなければ新しくテキストファイルを作成し，テキストデータを保存する．同じ名前のテキストファイルがある場合，そのテキストファイルの最後の行にテキストデータを追加し保存する．

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を整理したものである．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 の，tags はタグ 5 個，caption は取得したキャプションを表している．タグは上位 5 個にマウスが入ることは少な

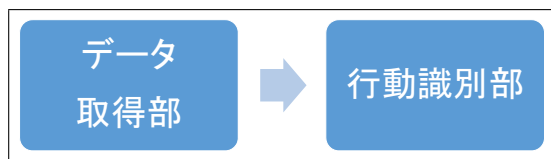


図 4.1: システム全体図

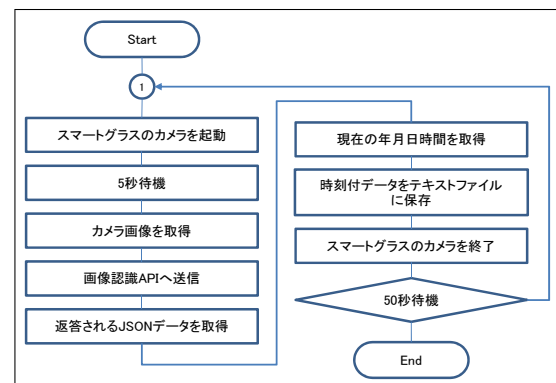


図 4.2: アプリケーションのフローチャート

表 4.1: 5回取得を行った結果

取得回数	tags	caption
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

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。

§ 4.2 省電力化

開発したアプリケーションのバッテリー消費に関しての工夫について述べる。図 4.3 は、スマートグラスのカメラ機能を起動させたまま 1 分ごとにデータ取得を続けるアプリケーションのフローチャートである。この時、カメラを起動させたままだと 2 時間程度でスマートグラスのバッテリーがなくなってしまう。充電不可能な外出先でのデータ取得のため、少しでも稼働時間を伸ばす必要がある。

この問題に対し、本研究ではカメラの起動・終了にかかるバッテリー消費よりも、連続起動の方がバッテリー消費が大きいと考え、データ取得後にカメラ機能を終了するようにプログラムに組み込んでいる（図 4.4 参照）。カメラ終了をプログラムに組み込むことにより、バッテリーのみで 3 時間から 3 時間半程度稼働することができた。なお、MOVERIO の標準的な駆動時間は約 6 時間¹⁰となっている。

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。理由として、カメラを起動するのに少なからず時間がかかるため、起動後すぐに撮影を行いカメラ画像データの取得は難しいためである。また、その後画像認識 API の応答を得るまでおよそ 5 秒程度かかるため、約一分ごとにデータを

¹⁰<http://www.epson.jp/products/moverio/bt300/spec.htm>

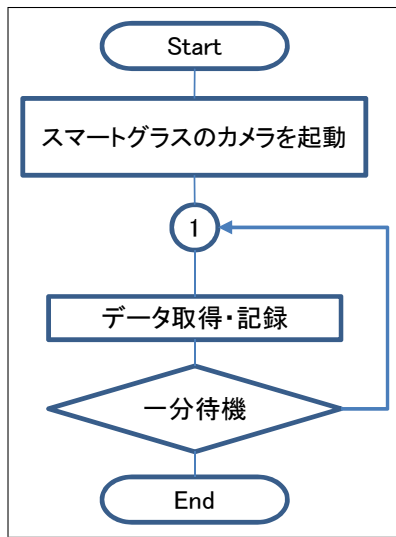


図 4.3: 省電力化前

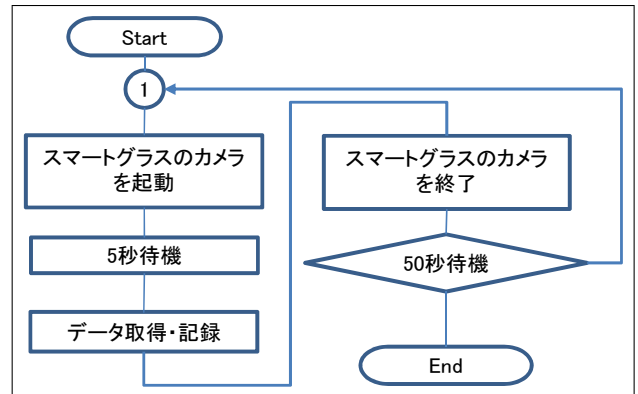


図 4.4: 省電力化後

取得できるようにしている。

§ 4.3 周期性の検出

データ取得部の次に行う，行動識別部について述べる．行動識別部では，データ取得部で得たデータを整理し，KH Coder と R を用いて多変量解析を行い，ライフログデータの周期性を検出する．

開発したアプリケーションは，MOVERIO のカメラ画像を画像認識 API に送信し，約一分ごとに以下のテキストデータを取得，記録する．

```
[2018-01-28 10:30:12]{ "description":{ "tags":["indoor","laptop","table","computer","sitting","top","open","desk","white","keyboard","room","man","mouse","plate","laying","bed","playing"], "captions":[{"text":"an open laptop computer sitting on a table","confidence":0.95249546527278339}]},"requestId":"21b3c022-3d1b-4bc1-9cc8-df210d1f2094","metadata":{"height":720,"width":1280,"format":"Jpeg"}}
```

多変量解析を行う前に，前処理としてテキストデータのうち多変量解析に必要なデータのみを抽出する．本研究ではタグをに 5 個，キャプションを confidence を 1 つ抽出する．抽出した下記のテキストデータを解析を行いたい時間分テキストファイルに保存する．

```
an open laptop computer sitting on a table,indoor,laptop,table,computer,sitting
```

保存したテキストファイルを KH Coder を使用して，多変量解析する．この時，テキストデータの時系列から周期性を検出するために SOM を使用する．まず，KH Coder で SOM を作成する．このときクラスター数はクラスター解析などの結果から決定できる．KH Coder で SOM を作成した際に取得できる R ファイルの内容を，ライフログデータの時系列関係がわかるように書き換える必要がある．

R ファイルを読み込み，som パッケージを用いて SOM を出力する前に，以下のコマンドを追加する．追加することで，抽出語どうしの関係性を示す SOM ではなく，文章どうし，

本研究では一分ごとのライフログデータどうしの関係性を示す SOM を出力できる.

```
d <- t(d)
rownames(d) <- 1:nrow(d)
```

また, 本研究ではライフログデータの時系列関係をより視覚的に理解するため, R ファイルの最後に以下のコマンドを追加する. 以下のコマンドを追加することで, 出力された SOM データが格納されたデータフレーム `points` がプロットされた `id` 上に線分のみを上書きできる.

```
par(new=T)
plot(points[,1],points[,2],type="c",col="色指定")
```

なお, 二種類のデータを比較する SOM を作成する場合は, 一つのテキストファイルに二種類のテキストファイルをまとめ以下のコマンドを追加する. 以下のコマンドを追加することで, データフレーム `points` を二種類のデータに戻し, 各々の色で線分を上書きできる.

```
points1<-head(points,n総ライフログデータ=/2)
par(new=T)
plot(points1[,1],points1[,2],type="c",col="色指定 1")

points2<-tail(points,n総ライフログデータ=/2)
par(new=T)
plot(points2[,1],points2[,2],type="c",col="色指定 2")
```

これらの R コマンドを使用して, ライフログデータの時系列を表示できる SOM を出力する. 出力された SOM から行動パターンの類似性やイベント性を検出する.

シミュレーション結果ならびに考察

開発したアプリケーションを実際に使用して、ライフログデータを取得する。また、取得したデータを多変量解析を用いて、行動パターンの類似性やイベント性を検出する。

ライフログデータの取得日は2018年1月27日と28日の10時30分から13時30分の180分である。デバイスの充電が100%である状態から充電が切れるまで取得を行ったため取得時間は180分となっている。なお、おおよそ一分に一回データを取得したが、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。27日に取得したデータをデータ1、28日に取得したデータをデータ2とする。

図5.1にデータ1とデータ2のタイムスケジュールを示す。データ1は学校でデスクトップPCで作業を行い、外出するというライフログデータである。データ2は自宅でノートPCでの作業と食事を行うというライフログデータである。データ1、データ2共に、取得したテキストデータの中から confidence の高いタグ上位5個とキャプションを一行とした190行のテキストファイルを解析に使用する。

データ1とデータ2の比較を行いやすくするため、データ1とデータ2を一つのcsvファイルにしたものをデータ3とする（表5.1参照）。この時 label 列はデータ1とデータ2の区切りを格納してある。行番号1から190がデータ1であり、191から380がデータ2がとなっている。データ3を用いることで、データ1とデータ2の多変量解析結果を一つの結果上で確認できる。

KH Coder を用いて、取得したテキストファイルの前処理として自然言語処理を行い単語を抽出する。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。理由として、データ1、データ2共に抽出語を20個前後にし、プロットされる抽出語を減らすことで解析結果を読み取りやすくするためである。また、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ことが多く、コンピューターが置いてある、という状態を an open laptop computer sitting on a table というように画像認識APIが返答してしまうため、本研究では sitting という状態がノイズになってしまう。そのため品詞を絞り、ノイズを減らすこととした。取得したデータから KH Coder で階層的クラスター分析、MDS、対応分析、共起ネットワーク、SOM を出力する。

まず、データ1とデータ2のクラスター分析を行う。図5.2はデータ1から作成したクラ

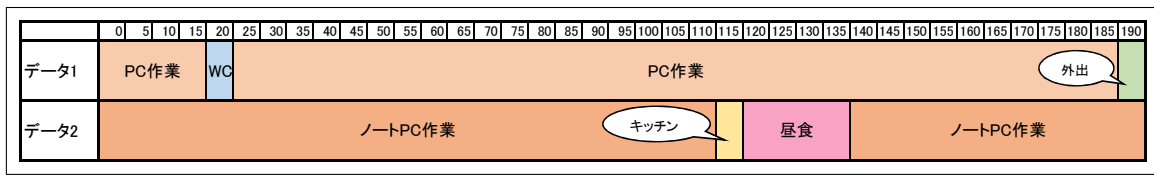


図 5.1: データ 1 とデータ 2 のタイムスケジュール

表 5.1: データ 3

	label	textdata
1	data1	a desk with a computer monitor,indoor,electronics,computer,monitor,table
2	data1	a desk with a computer monitor,indoor,monitor,computer,table,desk
3	data1	a desk with a computer monitor,indoor,computer,table,monitor,desk
4	data1	a desk with a computer monitor,indoor,monitor,table,computer,desk
⋮	⋮	⋮
377	data2	a stack of flyers on a table,indoor,table,top,sitting,desk
378	data2	a stack of flyers on a table,indoor,table,top,sitting,desk
379	data2	a stack of flyers on a table,indoor,table,top,sitting,desk
380	data2	a stack of flyers on a table,indoor,table,top,sitting,desk

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 の併合標準よりクラスター数 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 のままとした。データ 1 のクラスター分析より、赤のクラスターは computer や keyboard が含まれることから PC 作業であり出現回数もその他のクラスターに比べると最も多いことがわかる。青のクラスターは car や snow が含まれていること、outdoor という単語が含まれていることから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 はデータ 2 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 の併合標準より、クラスター数は 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 のままとした。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る。データ 1 と比較すると、PC 作業を表す単語の中に desktop や keyboard は含まれず、laptop が含まれていることからノート PC での作業を確認できる。ピンクのクラスターは food や plate から食事を表し、緑のクラスターはキッチンを表していると考ええるが、PC で動画を見ながら食事を行っていたため、食べ物以外の物体との関係性が強く表され、自室の私物も含まれてしまっている。食事していることをより正確にライフログデータとして取得するには、食べ物をなるべく視界に入れてデータを取得しなくてはいけないのではないかと考えた。また、図 5.2 と図 5.4 を比較すると、データ 2 の自宅の方が視界に入る物体が多いことから bottle や flyer など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。

次に、データ 1 とデータ 2 の MDS を行う。図 5.6 はデータ 1 から作成した MDS である。クラスター分析より、クラスター数は 4 とした。PC 作業に関する computer や desk という抽出語から構成されている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

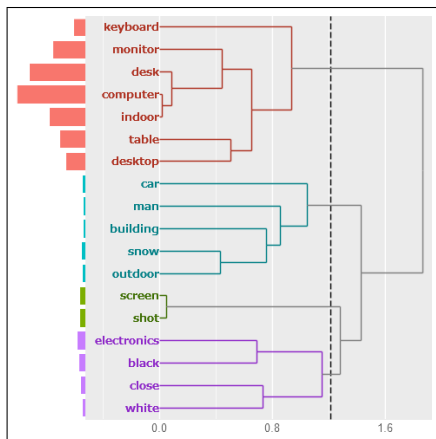


図 5.2: データ 1 のクラスター分析

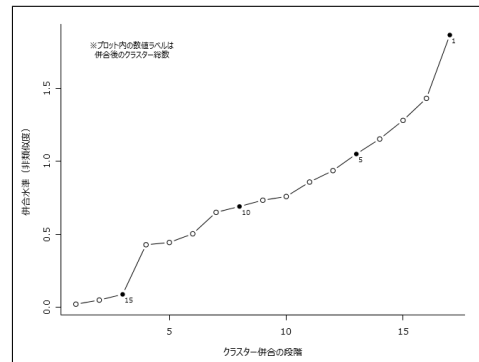


図 5.3: データ 1 の併合標準

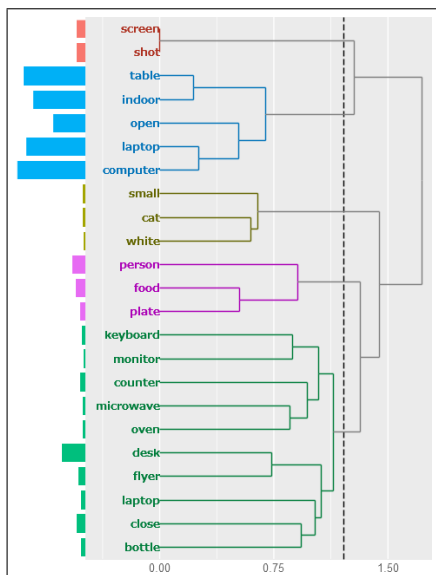


図 5.4: データ 2 のクラスター分析

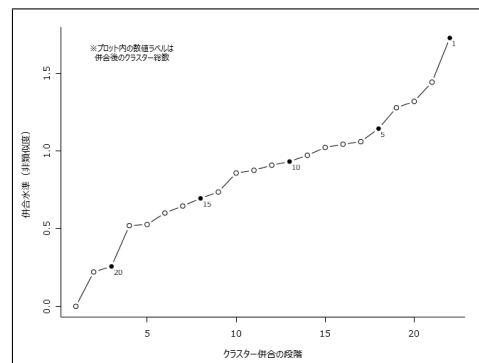


図 5.5: データ 2 の併合標準

れていることとクラスター分けから行動がはっきり分かれていることがわかる。なお、クラスター数を 3 にして出力を行うとクラスター 03 と 04 が一つのクラスターになったため、クラスター 03 は 02 よりも 04 に近いものとする。

図 5.7 はデータ 2 から作成した MDS である。クラスター分析より、クラスター数は 5 とした。computer や screen から構成されるクラスター 01 と desk や flyer から構成される 02 は PC 作業という行動を示すため、近い位置に配置されていることがわかる。クラスター 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のではないかと推測できるプロットとなっている。01 から少し離れた 04 に oven や microwave という抽出語があるため、オーブンや電子レンジを使用したことなどが考えられる。また、クラスター 05 は 01～04 より少し離れているためノイズではないかと考えられる。図 5.6 と図 5.7 を比較すると、データ 2 のほうがクラスター間のプロットの距離が近いことがわかる。このことから、実際に似たような行動を複

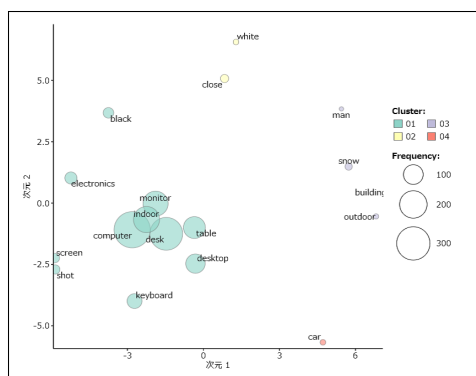


図 5.6: データ 1 の MDS

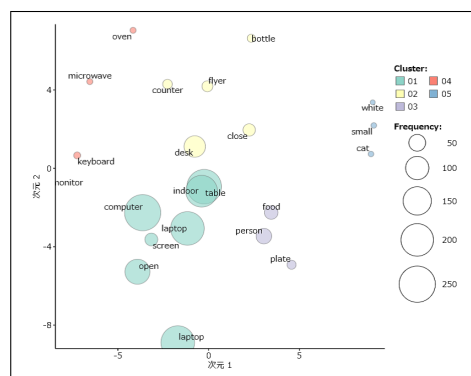


図 5.7: データ 2 の MDS

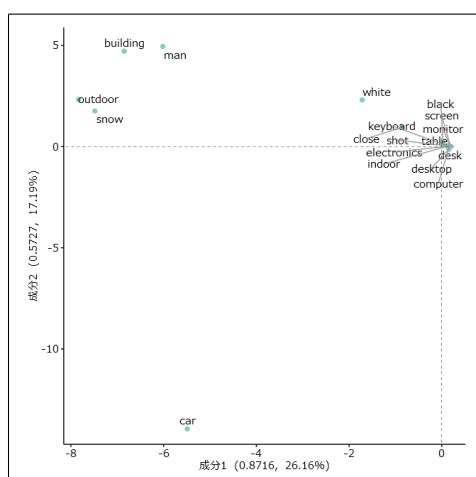


図 5.8: データ 1 の対応分析

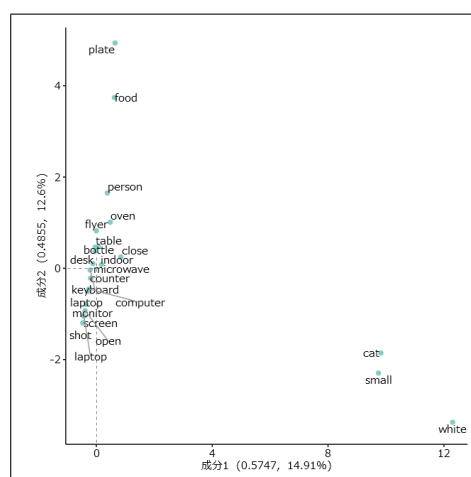


図 5.9: データ 2 の対応分析

数行っているか、行動を行っている際の視界情報が多いのではないかと考えられる。データ 1 もデータ 2 もクラスター 01 は PC 作業に関する抽出語で構成されているため、PC 作業は類似した行動ではないかと考えられる。クラスター 01 が一番大きく、他のクラスターと大きく差がある点は類似しているが、他のクラスターを構成する抽出語の相違からイベント性を検出できる。

次に、データ 1 とデータ 2 の対応分析を行う。図 5.8 はデータ 1 から作成した対応分析である。図 5.8 より、computer や keyboard で構成される行動である PC 作業を行っていることが最も多く、データ 1 内で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を構成する抽出語であると考えられる。また、building や outdoor から室外での行動であることが考えられる。

図 5.9 はデータ 2 から作成した対応分析である。図 5.8 と図 5.9 の比較すると、データ 1 もデータ 2 も PC 作業を中心に行っているが、データ 2 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。このことから食事をとったことが推測できる。

さらに、データ 1 とデータ 2 の関係性を同時に出力することができるため、データ 3 の対応分析を行う。5.10 より、データ 1、データ 2 共に同じくらい出現している抽出語、つまり

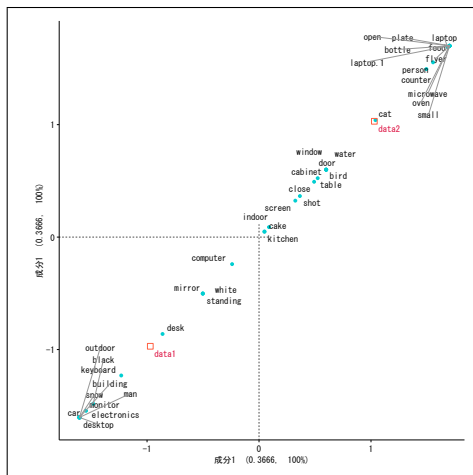


図 5.10: データ 3 の対応分析

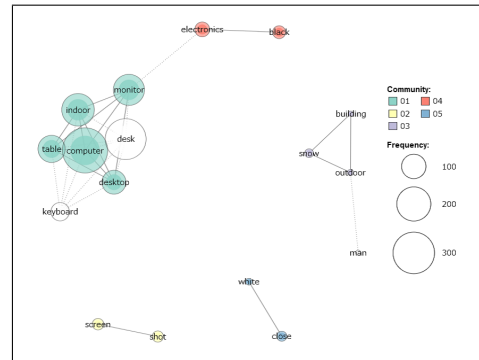


図 5.11: データ 1 の共起ネットワーク

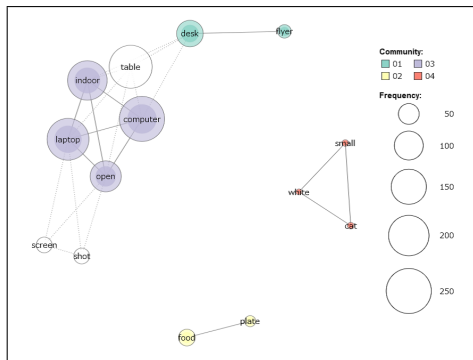


図 5.12: データ 2 の共起ネットワーク

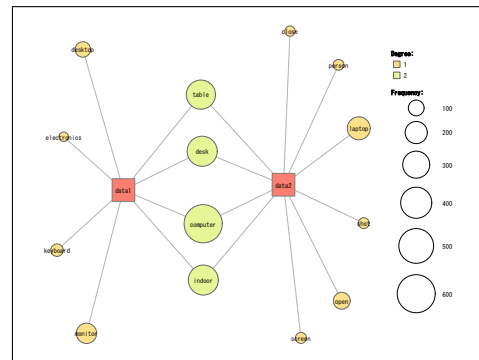


図 5.13: データ 3 の共起ネットワーク

特徴的ではない抽出語として indoor や computer が出現している。データ 1 からみて、データ 2 に含まれる table 等は関係性が近いが、food 等は関係性がないため特徴的であるように出力されている。同じようにデータ 2 からみて、データ 1 に含まれる snow 等は特徴的な語となっている。この比較より、データ 1 とデータ 2 には類似する行動もあることがわかる。

次に、データ 1 とデータ 2 の共起ネットワークを行う。図 5.11 はデータ 1 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写している。図 5.11 より、computer や monitor など 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。また、electronics と black とも、クラスターは違っているが共起関係があることがわかる。図 5.12 はデータ 2 から作成した共起ネットワークであり、図 5.11 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、クラスターも少なくなっていることがわかる。

さらに、データ 1 とデータ 2 の共起関係性を同時に出力するため、データ 3 の共起ネットワークを行う。5.13 より、データ 1 とデータ 2 はともに table, desk, computer, indoor という抽出語と共起関係があり、両方とも Jaccard 係数が 0.2 以上の強い共起関係をもつのは PC 作業を表す抽出語であり、類似性のある行動が確認できる。

最後に今までの解析を踏まえてライフログデータの時系列を可視化するため、SOM を作

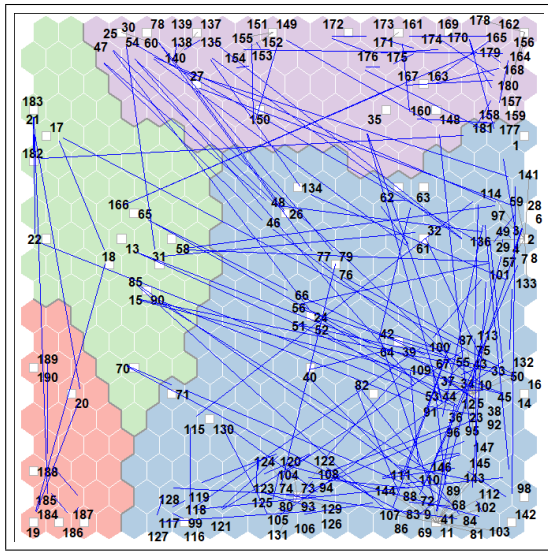


図 5.14: データ 1 の SOM

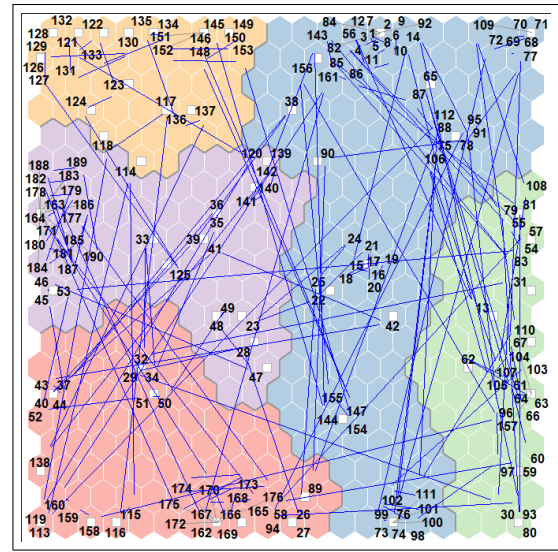


図 5.15: データ 2 の SOM

成する。クラスター数はデータ 1 が 4，データ 2 が 5 となっている。図 5.14 からデータ 1 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 5.15 からデータ 2 は食事関係を表す黄色のクラスター以外は，クラスター間を大きくまたぐ線などもあり，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。

また，データ 3 の SOM を作成し，データ 1 とデータ 2 の時系列の類似性を比較する。データは赤色の線分，データ 2 は青色の線分で示す。クラスター数は少ない方に合わせ 4 とした。また，データ 3 の SOM を作成するソースコードはソースコード A.2 に示す。図 5.16 より，データ 1 とデータ 2 の時系列は類似性が低いことがわかる。理由として，同じ PC 作業であっても視界に入る物体はデスクトップ PC とノート PC であるため同じ行動の中でも視界に写る物体が違うためであると考えられる。視界にある物体から行動の区別ができていたことがわかる。また，赤い線と青い線が両方ともつながっている時間のテキストを確認すると，a screen shot of a computer や a close up of a computer などの PC 作業の中で出現したノイズのようなテキストであった。

階層的クラスター分析，MDS，対応分析，共起ネットワークの解析から，データを構成する行動の検出，類似する行動とそうではないイベント性のある行動を検出することができた。また，SOM の解析から，同じ行動でも視界に写る物体の違いから行動の類似性やイベント性を検出できた。この結果から，同じ行動でも使用する場所や物体の変化によって別行動として認識させることができるため，GPS を使用せず，ライフログデータに位置情報を付加できると考えられる。よって，個人情報保護に着目し取得したライフログデータから類似性やイベント性を検出できたと考える。

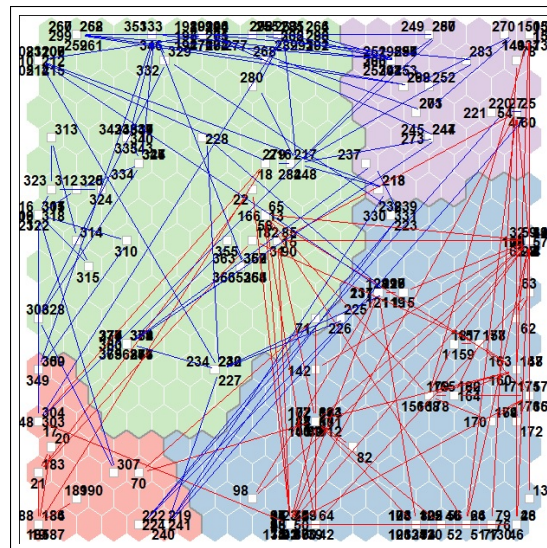


図 5.16: データ 3 の SOM

結論と今後の課題

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。

結論として、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。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。

今後の課題として、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。

謝辞

本研究を遂行するにあたり，多大なご指導と終始懇切丁寧なご鞭撻を賜った富山県立大学電子・情報工学科の奥原浩之教授に深甚な謝意を表します．最後になりましたが，多大な協力をして頂いた研究室の同輩諸氏に感謝致します．

2018 年 2 月

福島 瑞希

参考文献

- [1] 相澤清晴, “ライフログ”, 映像情報メディア学会誌, Vol. 63, No. 4, pp. 445–448, 2009.
- [2] “「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.
- [3] 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.
- [4] 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.
- [5] 角田宏貴, 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, 2014.
- [6] 矢野裕司, 横井健, 橋山智訓, “行動辞書を利用した Twitter からの行動抽出”, 情報科学技術フォーラム講演論文集, Vol. 11, No. 4, pp. 51–56, 2012.
- [7] 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.
- [8] 入江英嗣, 森田光貴, 岩崎央, 千竈航平, 放地宏佳, 小木真人, 桧原裕大, 芝星帆, 眞島一貴, 努吉永, “AirTarget: 光学シースルー方式 HMD とマーカレス画像認識による高可搬性実世界志向インタフェース”, 情報処理学会論文誌, Vol. 55, No. 4, pp. 1415–1427, 2014.
- [9] 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.
- [10] 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>, 閲覧日 2018,1,4.
- [11] 大雄治, 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.
- [12] 小林亜令, 岩本健嗣, 西山智, “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, 2008.
- [13] 寺田努, “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.
- [14] 貴志一樹, 山崎俊彦, 相澤清晴, “机上行動のライフログのための行動認識”, 映像情報メディア学会年次大会講演予稿集, Vol. 2014, , 2014.

- [15] 前川卓也, 柳沢豊, 岸野泰恵, 石黒勝彦, 亀井剛次, 櫻井保志, 岡留剛, “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, 研究報告ユビキタスコンピューティングシステム (UBI), 2010.
- [16] 樋口耕一, “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, 2004.
- [17] 佐野香織, 李在鎬, “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, 2007.
- [18] “KH Coder を用いた研究事例のリスト”, <http://khc.sourceforge.net/bib.html>, 閲覧日 2018,1,7.
- [19] 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.
- [20] “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.
- [21] “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.
- [22] 吉原一紘, 徳高平蔵, “クラスター分析の概要”, *Journal of Surface Analysis*, Vol. 21, No. 1, pp. 10–17, 2014.
- [23] 李美龍, 田中恒也, 成田吉弘, “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.
- [24] 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』のケインズの要素-テキストマイニングを加味した量的・質的分析-”, 2017.
- [25] 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.
- [26] Joseph B Kruskal, “Multidimensional scaling by optimizing goodness of fit to a non-metric hypothesis”, *Psychometrika*, Vol. 29, No. 1, pp. 1–27, 1964.
- [27] “Jaccard 係数の計算式と特徴 (1)”, <https://www.slideshare.net/khcoder/jaccard1>, 閲覧日 2018,2,3.
- [28] 中山慶一郎, “< 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, 2009.
- [29] 田中京子, “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, 2014.

- [30] “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.
- [31] 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.
- [32] 増田正, 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, 2012.
- [33] T. KOHONEN, “Self-organized formation of topologically correct feature map”, *Biol. Cybern.*, Vol. 43, pp. 59–69, 1982.
- [34] 岡晋之介, “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>, 閲覧日 2018,1,7.
- [35] “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.
- [36] 勝治宏基, 米澤拓郎, 中澤仁, 高汐一紀, 徳田英幸ほか, “Synchrometer: ライフログを利用した日常行動における他者との類似度生成”, 研究報告ユビキタスコンピューティングシステム (UBI), Vol. 2013, No. 17, pp. 1–7, 2013.

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 A.1 をしめす.

ソースコード A. 1: app.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 som : MonoBehaviour
11 {
12     private float captureIntervalSeconds = 50.0f;
13     private float captureIntervalSeconds2 = 5.0f;
14     public Text gtext;
15     Dictionary<string, string> headers;
16     private int Width = 1280;
17     private int Height = 720;
18     private int FPS = 30;
19     private WebCamTexture webcamTexture;
20     private Color32[] color32;
21     private string tagsStg;
22     private string reportFileName2 = "long_report.txt";
23     public bool addDateTime = true;
24
25     void Start ()
26     {
27         Screen.sleepTimeout = SleepTimeout.NeverSleep;
28         StartCoroutine ("Sample");
29     }
30
31     public IEnumerator Sample ()
32     {
33         WebCamDevice[] devices = WebCamTexture.devices;
34         WebCamDevice userCameraDevice = WebCamTexture.devices [0];
35         webcamTexture = new WebCamTexture (userCameraDevice.name,
36             Width, Height, FPS);
37         webcamTexture.Play ();
38         Debug.Log ("webcamTexture");
39
40         yield return new WaitForSeconds (captureIntervalSeconds2);
41         color32 = webcamTexture.GetPixels32 ();
42         Texture2D texture = new Texture2D (webcamTexture.width,
43             webcamTexture.height);
44         texture.SetPixels32 (color32);
45         texture.Apply ();
46         byte[] jpg = texture.EncodeToJPG ();
```

```

45     string VISIONKEY = "ba7982e18b4943d18024749aca8031fb";
46     var uri = "https://westus.api.cognitive.microsoft.com/vision/
        v1.0/describe";
47
48     string responseData;
49     var headers = new Dictionary<string, string> () {
50         { "Ocp-Apim-Subscription-Key", VISIONKEY },
51         { "Content-Type", "application/octet-stream" }
52     };
53
54     WWW www = new WWW (uri, jpg, headers);
55     yield return www;
56     responseData = www.text;
57     var data = JsonUtility.FromJson<SumDataFromJson> (responseData);
58
59     DateTime dt = DateTime.Now;
60     string text2 = dt.ToString (" [yyyy-MM-dd_HH:mm:ss] ") +
        responseData.ToString () + "\n";
61     string outfile2 = reportFileName2;
62
63     if (addDateTime) {
64         string file2 = Path.GetFileNameWithoutExtension (
            reportFileName2);
65         string ext2 = Path.GetExtension (reportFileName2);
66         outfile2 = file2 + "_" + dt.ToString ("yyyyMMdd") +
            ext2;
67     }
68
69     SaveText (text2, Path.Combine (Application.persistentDataPath,
        outfile2));
70     MonoBehaviour.Destroy (texture);
71     MonoBehaviour.Destroy (webcamTexture);
72     color32 = null;
73     StartCoroutine ("StopRunTimeTemp");
74 }
75
76 public IEnumerator StopRunTimeTemp ()
77 {
78     webcamTexture.Stop ();
79     gtext.text = tagsStg;
80     yield return new WaitForSeconds (captureIntervalSeconds);
81     StartCoroutine ("Sample");
82 }
83
84 public static bool SaveText (string text, string path)
85 {
86     try {
87         using (StreamWriter writer = new StreamWriter (path, true))
88         {
89             writer.Write (text);
90             writer.Flush ();
91             writer.Close ();
92         }
93     } catch (Exception e) {
94         Debug.Log (e.Message);
95         return false;
96     }
97     return true;

```

A.2 時系列 SOM を作成するソースコード

データ3の時系列SOMを作成するソースコードA.2をしめす.

ソースコード A. 2: som.r

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```

408 d <- t(d)
409 rownames(d) <- 1:nrow(d)
410 # SOM
411 library(som)
412 somm <- som(
413     d,
414     n_nodes,
415     n_nodes,
416     topol="hexa",
417     rlen=c(rlen1,rlen2)
418 )
419
420 word_labs <- rownames(d)
421 n_words <- length(word_labs)
422
423
424 color_universal_design <- 1
425 # END: DATA
426
427 cex <- 1
428 text_font <- 2
429 if_cls <- 1
430 n_cls <- 4
431 if_plothex <- 1
432
433 # n_nodes <- 20
434 # rlen1 <- 1000
435 # rlen2 <- 200000
436
437 row2coods <- NULL
438 eve <- 0
439 for (i in 0:(n_nodes - 1)){
440     for (h in 0:(n_nodes - 1)){
441         row2coods <- c(row2coods, h + eve, i)
442     }
443     if (eve == 0){
444         eve <- 0.5
445     } else {
446         eve <- 0
447     }
448 }
449 row2coods <- matrix( row2coods, byrow=T, ncol=2 )
450
451
452 if ( if_cls == 1 ){
453     library( RColorBrewer )
454
455     if (
456         ( as.numeric( R.Version()$major ) >= 3 )
457         && ( as.numeric( R.Version()$minor ) >= 1.0)
458     ){ # >= R 3.1.0
459         hcl <- hclust( dist(somm$code,method="euclidean"), method="ward
460             .D2" )
461     } else { # <= R 3.0
462         hcl <- hclust( dist(somm$code,method="euclidean")^2, method="
463             ward" )
464     }
465
466     colors <- NULL
467     if (n_cls <= 9){

```

```

466         pastel <- brewer.pal(9, "Pastel1")
467         # pastel[6] = brewer.pal(9, "Pastel1")[9]
468         # pastel[9] = brewer.pal(9, "Pastel1")[6]
469         pastel[6] = "gray91"
470         pastel[9] = "#F5F5DC" # FAF3C8 F7F1C6 EEE8AA F0E68C
471         colors <- pastel[cutree(hcl,k=n_cls)]
472     }
473     if (n_cls > 9) {
474
475         library(colorspace)
476         new_col <- order( runif(n_cls) )
477         colors <-
478             rainbow_hcl(n_cls, start=20, end=340, l=92, c=20)[
479                 #terrain_hcl(n_cls, c = c(35, 5), l = c(85, 95), power = c(0.5,1)
480                 )/
481                 new_col[cutree(hcl,k=n_cls)]
482     }
483 } else {
484     colors <- rep("gray90", n_nodes^2)
485 }
486 labcd <- NULL
487
488 plot_mode <- "color"
489
490
491 par(mai=c(0,0,0,0), mar=c(0,0,0,0), omi=c(0,0,0,0), oma =c(0,0,0,0) )
492
493 plot(
494     NULL,NULL,
495     xlim=c(0,n_nodes-0.5),
496     ylim=c(0,n_nodes-1),
497     axes=F,
498     frame.plot=F
499 )
500
501 if (if_plothe == 1){
502     a <- 0.333333333333
503 } else {
504     a <- 0.5
505 }
506 b <- 1-a
507
508 color_pte <- "gray70"
509 cls_lwd <- 2
510
511 if ( plot_mode == "gray"){
512     color_act <- rep("white",n_nodes^2)
513     if_points <- 1
514     w_lwd <- 1
515     cls_lwd <- 2.25
516     color_cls <- "gray35"
517     color_line <- "gray50"
518     color_pte <- "gray40"
519     color_ptf <- "gray85"
520 }
521 if ( plot_mode == "color" ) {
522     color_act <- colors
523     color_line <- "white"
524     if_points <- 1

```

```

525     w_lwd <- 1
526     if (n_cls > 9) {
527         color_cls <- "gray45"
528     } else {
529         color_cls <- "gray60"
530     }
531     color_ptf <- "white"
532 }
533 if ( plot_mode == "freq" ){
534     color_act <- somm$code.sum$nobs;
535     if (max(color_act) == 1){
536         color_act <- color_act * 3 + 1;
537     } else {
538         color_act <- color_act - min(color_act)
539         color_act <- round( color_act / max(color_act) * 6 ) + 1
540         #color_act[color_act==7] <- 6
541     }
542     color_seed <- brewer.pal(6,"GnBu")
543     #color_seed <- brewer.pal(6,"YlOrRd")
544     color_seed <- c("white", color_seed)
545     color_act <- color_seed[color_act]
546
547     color_line <- "gray70"
548     if_points <- 0
549     w_lwd <- 1
550     color_cls <- "gray45"
551     color_ptf <- "white"
552 }
553 if ( plot_mode == "umat" ){
554
555
556     dist_u <- NULL
557
558     dist_m <- as.matrix( dist(somm$code, method="euclid") )
559
560     for (i in 0:(n_nodes - 1)){
561         for (h in 0:(n_nodes - 1)){
562             cu <- NULL
563             n <- 0
564
565             if (h != n_nodes - 1){
566                 cu <- c(
567                     cu,
568                     dist_m[
569                         h + i * n_nodes + 1,
570                         h + 1 + i * n_nodes + 1
571                     ]
572                 )
573             }
574
575             if (h != 0){
576                 cu <- c(
577                     cu,
578                     dist_m[
579                         h + i * n_nodes + 1,
580                         h - 1 + i * n_nodes + 1
581                     ]
582                 )
583             }
584

```

```

585     if (i != n_nodes - 1){
586         if (h %% 2 == 0){
587             cu <- c(
588                 cu,
589                 dist_m[
590                     h + i * n_nodes + 1,
591                     h + ( i + 1 ) * n_nodes + 1
592                 ]
593             )
594         } else {
595             if (h != n_nodes - 1){
596                 cu <- c(
597                     cu,
598                     dist_m[
599                         h + i * n_nodes + 1,
600                         h + 1 + ( i + 1 ) * n_
601                             nodes + 1
602                     ]
603                 )
604             }
605         }
606     }
607     if (i != 0){
608         if (h %% 2 == 0){
609             cu <- c(
610                 cu,
611                 dist_m[
612                     h + i * n_nodes + 1,
613                     h + ( i - 1 ) * n_nodes + 1
614                 ]
615             )
616         } else {
617             if (h != n_nodes - 1){
618                 cu <- c(
619                     cu,
620                     dist_m[
621                         h + i * n_nodes + 1,
622                         h + 1 + ( i - 1 ) * n_
623                             nodes + 1
624                     ]
625                 )
626             }
627         }
628     }
629     if (i != n_nodes - 1){
630         if (h %% 2 == 0){
631             if (h != 0){
632                 cu <- c(
633                     cu,
634                     dist_m[
635                         h + i * n_nodes + 1,
636                         h - 1 + ( i + 1 ) * n_
637                             nodes + 1
638                     ]
639                 )
640             } else {

```

```

641             cu <- c(
642                 cu,
643                 dist_m[
644                     h + i * n_nodes + 1,
645                     h + ( i + 1 ) * n_nodes + 1
646                 ]
647             )
648         }
649     }
650
651     if (i != 0){
652         if (h %% 2 == 0){
653             if (h != 0){
654                 cu <- c(
655                     cu,
656                     dist_m[
657                         h + i * n_nodes + 1,
658                         h - 1 + ( i - 1 ) * n_
659                             nodes + 1
660                     ]
661                 )
662             } else {
663                 cu <- c(
664                     cu,
665                     dist_m[
666                         h + i * n_nodes + 1,
667                         h + ( i - 1 ) * n_nodes + 1
668                     ]
669                 )
670             }
671         }
672         dist_u <- c(dist_u, median(cu) )
673     }
674 }
675
676 print( summary(dist_u) )
677
678 dist_u <- dist_u - min(dist_u)
679 dist_u <- round( dist_u / max(dist_u) * 100 ) + 1
680
681 if (color_universal_design == 0){
682     color_act <- cm.colors(101)[dist_u]
683     color_line <- "gray70"
684     color_cls <- "gray45"
685 } else {
686     library(RColorBrewer)
687     if (T){
688         col_seed <- brewer.pal(9, "GnBu")
689         myPalette <- colorRampPalette( col_seed )
690         color_act <- myPalette(101)[dist_u]
691         color_act <- adjustcolor(color_act, alpha=0.8)
692         color_line <- "white"
693         color_cls <- "gray30"
694     } else {
695         col_seed <- rev(brewer.pal(9, "RdYlBu"))
696         myPalette <- colorRampPalette( col_seed )
697         color_act <- myPalette(101)[dist_u]
698         color_act <- adjustcolor(color_act, alpha=0.8)

```

```

699             color_line <- "gray50"
700             color_cls <- "gray25"
701         }
702     }
703
704     #color_line <- "gray70"
705     if_points <- 1
706     w_lwd <- 1
707     #color_cls <- "gray45"
708     color_ptf <- "white"
709 }
710
711
712 for (i in 1:n_nodes^2){
713     x <- row2coods[i,1]
714     y <- row2coods[i,2]
715
716     polygon(
717         x=c( x + 0.5, x + 0.5, x, x - 0.5, x - 0.5, x ),
718         y=c( y + a, y - a, y - b, y - a, y + a, y + b ),
719         col=color_act[i],
720         border="white",
721         lty=0,
722     )
723 }
724
725
726 for (i in 0:(n_nodes - 1)){
727     for (h in 0:(n_nodes - 2)){
728         if ( colors[h + i * n_nodes + 1] == colors[h + i * n_nodes + 2] ){
729             x <- h
730             y <- i
731             if ( y %% 2 == 1 ){
732                 x <- x + 0.5
733             }
734
735             segments(
736                 x + 0.5, y + a,
737                 x + 0.5, y - a,
738                 col=color_line,
739                 lwd=w_lwd,
740             )
741         }
742     }
743 }
744
745 for (i in 0:(n_nodes - 1)){
746     for (h in c(-1, n_nodes-1) ){
747         x <- h
748         y <- i
749         if ( y %% 2 == 1 ){
750             x <- x + 0.5
751         }
752         segments(
753             x + 0.5, y + a,
754             x + 0.5, y - a,
755             col=color_line,
756             lwd=w_lwd,
757         )
758     }

```

```

759     if ( y %% 2 == 0 ){
760         segments(
761             -0.5, y + a,
762             0 , y + 1 - a,
763             col=color_line,
764             lwd=w_lwd,
765         )
766         if ( y != 0 ){
767             segments(
768                 -0.5, y - a,
769                 0 , y - 1 + a,
770                 col=color_line,
771                 lwd=w_lwd,
772             )
773         }
774     } else {
775         if ( y != n_nodes - 1 ){
776             segments(
777                 n_nodes - 0.5, y + 1 - a,
778                 n_nodes , y + a,
779                 col=color_line,
780                 lwd=w_lwd,
781             )
782         }
783         segments(
784             n_nodes - 0.5, y - 1 + a,
785             n_nodes , y - a,
786             col=color_line,
787             lwd=w_lwd,
788         )
789     }
790 }
791
792 for (i in 0:(n_nodes - 2)){
793     for (h in 0:(n_nodes - 1)){
794         if (i %% 2 == 1){
795             chk <- 1
796         } else {
797             chk <- 0
798         }
799
800         if (
801             is.na(colors[h + i * n_nodes + 1]) == 1
802             || is.na(colors[h + chk + (i+1) * n_nodes + 1]) == 1
803             || h + chk == n_nodes
804         ){
805             next
806         }
807
808         if (
809             colors[h + i * n_nodes + 1]
810             == colors[h + chk + (i+1) * n_nodes + 1]
811         ){
812             x <- h
813             y <- i
814             if ( y %% 2 == 1 ){
815                 x <- x + 0.5
816             }
817
818             segments(

```

```

819         x, y + b,
820         x + 0.5, y + a,
821         col=color_line,
822         lwd=w_lwd,
823     )
824 }
825 }
826 }
827
828 for (i in 0:(n_nodes - 2)){
829     for (h in 0:(n_nodes - 1)){
830         if (i %% 2 == 0){
831             chk <- 1
832         } else {
833             chk <- 0
834         }
835         if (
836             is.na(colors[h + i * n_nodes + 1]) == 1
837             || is.na(colors[h - chk + (i+1) * n_nodes + 1]) == 1
838             || h - chk < 0
839         ){
840             next
841         }
842         if (
843             colors[h + i * n_nodes + 1]
844             == colors[h - chk + (i+1) * n_nodes + 1]
845         ){
846             x <- h
847             y <- i
848             if ( y %% 2 == 1 ){
849                 x <- x + 0.5
850             }
851
852             segments(
853                 x, y + b,
854                 x - 0.5, y + a,
855                 col=color_line,
856                 lwd=w_lwd,
857             )
858         }
859     }
860 }
861 }
862
863
864
865 for (i in 0:(n_nodes - 1)){
866     for (h in 0:(n_nodes - 2)){
867         if ( colors[h + i * n_nodes + 1] != colors[h + i * n_nodes + 2] ){
868             x <- h
869             y <- i
870             if ( y %% 2 == 1 ){
871                 x <- x + 0.5
872             }
873
874             segments(
875                 x + 0.5, y + a,
876                 x + 0.5, y - a,
877                 col=color_cls,
878                 lwd=cls_lwd,

```

```

879         )
880     }
881 }
882 }
883
884 for (i in 0:(n_nodes - 2)){
885     for (h in 0:(n_nodes - 1)){
886         if (i %% 2 == 1){
887             chk <- 1
888         } else {
889             chk <- 0
890         }
891
892         if (
893             is.na(colors[h + i * n_nodes + 1]) == 1
894             || is.na(colors[h + chk + (i+1) * n_nodes + 1]) == 1
895             || h + chk == n_nodes
896         ){
897             next
898         }
899
900         if (
901             colors[h + i * n_nodes + 1]
902             != colors[h + chk + (i+1) * n_nodes + 1]
903         ){
904             x <- h
905             y <- i
906             if ( y %% 2 == 1 ){
907                 x <- x + 0.5
908             }
909
910             segments(
911                 x, y + b,
912                 x + 0.5, y + a,
913                 col=color_cls,
914                 lwd=cls_lwd,
915             )
916         }
917     }
918 }
919
920 for (i in 0:(n_nodes - 2)){
921     for (h in 0:(n_nodes - 1)){
922         if (i %% 2 == 0){
923             chk <- 1
924         } else {
925             chk <- 0
926         }
927
928         if (
929             is.na(colors[h + i * n_nodes + 1]) == 1
930             || is.na(colors[h - chk + (i+1) * n_nodes + 1]) == 1
931             || h - chk < 0
932         ){
933             next
934         }
935
936         if (
937             colors[h + i * n_nodes + 1]

```

```

938         != colors[h - chk + (i+1) * n_nodes + 1]
939     ){
940         x <- h
941         y <- i
942         if ( y %% 2 == 1 ){
943             x <- x + 0.5
944         }
945
946
947     # fix for "wordlayout" function    segments(
948         x, y + b,
949         x - 0.5, y + a,
950         col=color_cls,
951         lwd=cls_lwd,
952     )
953     }
954 }
955 }
956
957 points <- NULL
958 sf <- 0.35
959 a <- a * sf;
960 b <- b * sf;
961 c <- 0.5 * sf;
962 for (i in 1:nrow(somm$visual)){
963     x <- somm$visual[i,1]
964     y <- somm$visual[i,2]
965     if ( y %% 2 == 1 ){
966         x <- x + 0.5
967     }
968     points <- c(points, x, y)
969 }
970 points <- matrix( points, byrow=T, ncol=2 )
971
972 if( if_points == 1 ){
973     if (F){
974         for (i in 1:nrow(points)){
975             x <- points[i,1]
976             y <- points[i,2]
977
978             polygon(
979                 x=c( x + c, x + c, x, x - c, x - c, x ),
980                 y=c( y + a, y - a, y - b, y - a, y + a, y + b ),
981                 col=color_ptf,
982                 border=color_pte,
983                 lty=1,
984             )
985         }
986     } else {
987         symbols(
988             points[,1],
989             points[,2],
990             squares=rep(0.35,length(points[,1])),
991             #circles=rep(0.2,length(points[,1])),
992             fg="gray70",
993             bg=color_ptf,
994             inches=F,
995             add=T,
996         )

```

```

997     }
998 }
999
1000 library(maptools)
1001 if (is.null(labcd) == 1){
1002     labcd <- pointLabel(
1003         x=points[,1],
1004         y=points[,2],
1005         labels=word_labs,
1006         doPlot=F,
1007         cex=cex,
1008         offset=0
1009     )
1010
1011     xorg <- points[,1]
1012     yorg <- points[,2]
1013     #cex <- 1
1014
1015     if ( length(xorg) < 300 ) {
1016         library(wordcloud)
1017
1018     filename <- tempfile()
1019     writeLines("wordlayout <- function (x, y, words, cex = 1, rotate90 = FALSE,
1020         xlim = c(-Inf,
1021             Inf), ylim = c(-Inf, Inf), tstep = 0.1, rstep = 0.1, ...)
1022     {
1023         tails <- "\"g|j|p|q|y\"
1024         n <- length(words)
1025         sdx <- sd(x, na.rm = TRUE)
1026         sdy <- sd(y, na.rm = TRUE)
1027         iterations <- 0
1028         if (sdx == 0)
1029             sdx <- 1
1030         if (sdy == 0)
1031             sdy <- 1
1032         if (length(cex) == 1)
1033             cex <- rep(cex, n)
1034         if (length(rotate90) == 1)
1035             rotate90 <- rep(rotate90, n)
1036         boxes <- list()
1037         for (i in 1:length(words)) {
1038             rotWord <- rotate90[i]
1039             r <- 0
1040             theta <- runif(1, 0, 2 * pi)
1041             x1 <- xo <- x[i]
1042             y1 <- yo <- y[i]
1043             wid <- strwidth(words[i], cex = cex[i], ...)
1044             ht <- strheight(words[i], cex = cex[i], ...)
1045             if (grepl(tails, words[i]))
1046                 ht <- ht + ht * 0.2
1047             if (rotWord) {
1048                 tmp <- ht
1049                 ht <- wid
1050                 wid <- tmp
1051             }
1052             isOverlaped <- TRUE
1053             while (isOverlaped) {
1054                 if (!.overlap(x1 - 0.5 * wid, y1 - 0.5 * ht, wid,
1055                     ht, boxes) && x1 - 0.5 * wid > xlim[1] && y1

```

```

1056         0.5 * ht > ylim[1] && x1 + 0.5 * wid < xlim[2]
1057         &&
1058         y1 + 0.5 * ht < ylim[2]) {
1059             boxes[[length(boxes) + 1]] <- c(x1 - 0.5 * wid
1060                 ,
1061                 y1 - 0.5 * ht, wid, ht)
1062             isOverlaped <- FALSE
1063         }
1064         else {
1065             theta <- theta + tstep
1066             r <- r + rstep * tstep/(2 * pi)
1067             x1 <- xo + sdx * r * cos(theta)
1068             y1 <- yo + sdy * r * sin(theta)
1069             iterations <- iterations + 1
1070             if (iterations > 500000){
1071                 boxes[[length(boxes) + 1]] <- c(x1 -
1072                     0.5 * wid,
1073                     y1 - 0.5 * ht, wid, ht)
1074                 isOverlaped = FALSE
1075             }
1076         }
1077     }
1078     print( paste("\ iterations: \", iterations) )
1079     result <- do.call(rbind, boxes)
1080     colnames(result) <- c("\ x\ ", "\ y\ ", "\ width\ ", "\ ht\ ")
1081     rownames(result) <- words
1082     result
1083 }
1084 ", filename)
1085 insertSource(filename, package="wordcloud", force=FALSE)
1086 nc <- wordlayout(
1087     labcd$x,
1088     labcd$y,
1089     word_labs,
1090     cex=cex * 1.25,
1091     xlim=c( par( "usr" )[1], par( "usr" )[2] ),
1092     ylim=c( par( "usr" )[3], par( "usr" )[4] )
1093 )
1094 xlen <- par("usr")[2] - par("usr")[1]
1095 ylen <- par("usr")[4] - par("usr")[3]
1096 segs <- NULL
1097 for (i in 1:length(word_labs) ){
1098     x <- ( nc[i,1] + .5 * nc[i,3] - labcd$x[i] ) / xlen
1099     y <- ( nc[i,2] + .5 * nc[i,4] - labcd$y[i] ) / ylen
1100     dst <- sqrt( x^2 + y^2 )
1101     if ( dst > 0.05 ){
1102         segs <- rbind(
1103             segs,
1104             c(
1105                 nc[i,1] + .5 * nc[i,3], nc[i,2] + .5 * nc[i,4],
1106                 xorg[i], yorg[i]
1107             )
1108         )
1109     }
1110 }

```

```

1111
1112         xorg <- labcd$x
1113         yorg <- labcd$y
1114         labcd$x <- nc[,1] + .5 * nc[,3]
1115         labcd$y <- nc[,2] + .5 * nc[,4]
1116     }
1117
1118
1119 }
1120
1121 if ( exists("segs" ) ){
1122     if ( is.null(segs) == F ){
1123         for (i in 1:nrow(segs) ){
1124             segments(
1125                 segs[i,1],segs[i,2],segs[i,3],segs[i,4],
1126                 col="gray60",
1127                 lwd=1
1128             )
1129         }
1130     }
1131 }
1132
1133 text(
1134     labcd$x,
1135     labcd$y,
1136     labels=word_labs,
1137     cex=cex,
1138     offset=0,
1139     font=text_font
1140 )
1141
1142 if ( exists("out_coord" ) == F ) {
1143     out_coord <- cbind(
1144         labcd$x / (n_nodes-0.5),
1145         labcd$y / (n_nodes-1)
1146     )
1147 }
1148
1149 points1<-head(points,n=190)
1150 par(new=T)
1151 plot(points1[,1],points1[,2],type="c",col="red")
1152
1153 points2<-tail(points,n=190)
1154 par(new=T)
1155 plot(points2[,1],points2[,2],type="c",col="blue")

```
