

千葉工業大学 社会システム科学部  
プロジェクトマネジメント学科  
平成 26 年度 卒業論文

ユーザー関係の分析による

Twitter コミュニティ抽出

Twitter community extraction by the user-affiliated analysis

ソフトウェア開発コース  
矢吹研究室

1142123 渡邊 雄大／Yudai Watanabe

指導教員印	学科受付印

## 目次

第1章	序論.....	1
1.1	本章の構成 .....	1
1.2	研究背景 .....	1
1.3	研究目的 .....	1
1.4	プロジェクトマネジメントとの関連性.....	1
1.5	論文構成 .....	2
第2章	SNS について .....	3
2.1	本章の構成 .....	3
2.2	SNS とは.....	3
第3章	TWITTER について .....	6
3.1	本章の構成 .....	5
3.2	Twitter とは.....	5
3.3	用語 .....	6
3.3.1	ユーザー .....	6
3.3.2	ユーザー名(スクリーン名).....	6
3.3.3	TwitterID.....	6
3.3.4	ツイート .....	6
3.3.5	タイムライン.....	6
3.3.6	フォロー .....	6
3.3.7	フォロワー.....	7
3.3.8	フレンド .....	7
3.3.9	メンション (@関連ツイート) .....	8
3.3.10	リツイート.....	9
3.3.11	お気に入り .....	10
3.3.12	リスト .....	10
3.3.13	ハッシュタグ .....	10
3.3.14	トレンド.....	10
3.3.15	非公開ツイート.....	10
3.4	参考文献 .....	11
第4章	ネットワーク抽出について .....	12

4.1	本章の構成 .....	12
4.2	API とは .....	12
4.3	TwitterAPI とは .....	13
4.3.1	TwitterAPI の使用方法 .....	14
4.3.2	REST API とは .....	14
4.3.3	REST API リスト .....	14
4.3.4	Streaming API とは .....	23
4.3.5	レートリミットとは .....	23
4.3.6	AccessToken とは .....	24
4.4	TwitterAPI を実際に使ってみる .....	25
4.5	MySQL とは .....	27
4.5.1	MySQL を実際に使ってみる .....	29
第 5 章	本論 .....	26
5.1	本章について .....	32
5.2	研究方法 .....	32
5.2.1	研究を行うための用意 .....	39
5.2.2	研究の手順 .....	41
5.2.3	実際のコード .....	45
5.3	参考文献 .....	53
第 6 章	結果・考察 .....	35
6.1	本章について .....	54
6.2	研究結果 .....	54
6.3	研究結果考察 .....	56
6.3.1	研究の展望 .....	57
6.4	研究方法に対する考察 .....	57

# 第 1 章

## 序論

## 1.1 本章の構成

本章では、本研究の背景・目的・プロジェクトマネジメントとの関連を記す。

## 1.2 研究背景

コミュニケーションツールとして、Social Networking Service (SNS) を使用している人はとても多くいる。その中でも Twitter は、SNS を代表する 1 つである。なぜなら Twitter はアクティブユーザー数が 2 億 3 千万人もいることだけでなく、ツイートと呼ばれるマイクロブログが一日平均で 5 億件も送信されているからだ (2014 年 9 月現在)。そのため Twitter は、調査する価値のある SNS の 1 つであると考えられる。

Twitter はツイートと呼ばれる短い文字列を投稿するためのサービスである。自分以外のユーザーのツイートを読むためには、そのユーザーのページにアクセスするか、そのユーザーをフォローする必要がある。フォローしているユーザーのツイートは、ひとまとめにされ、タイムラインを形成する。誰が誰をフォローしているかという情報 (フォロー関係) は、Twitter におけるユーザーのつながりの一つの表現である。仮に Twitter 上でユーザーがコミュニティを形成していたとすれば、フォロー関係にもそれが反映されていると思われる。そのコミュニティを抽出することができれば、フォローすべきユーザーの発見が容易になるなど、Twitter のユーザビリティが大きく向上することが期待される。

## 1.3 研究目的

検索したいユーザーの Twitter 上に持っているフォロー関係から、そのユーザーの持つ実際のコミュニティを Twitter の機能であるリツイートを解析することによって見つけ出す。さらに、この研究を行うことで、プロジェクトを円滑に行うための人的資源マネジメントとして活用することを目指す。

## 1.4 プロジェクトマネジメントとの関連性

本研究では、SNS のユーザビリティの向上を目指す研究である。この研究によってプロジェクト・メンバ間の交流を促進させ、プロジェクトのパフォーマンスの向上ができると考えている。そのため人的資源マネジメントへ貢献することが期待される。

## 1.5 論文構成

本稿では以下のような構成をとる．第 1 章では序論として研究背景，目的の解説，プロジェクトマネジメントとの関連性．第 2 章では SNS についての解説．第 3 章では研究対象である Twitter の基本事項の解説．第 4 章では TwitterAPI を使用したネットワーク抽出方法の解説．第 5 章では研究方法と実際に使用したプログラムについての解説．第 6 章では研究の結果と考察を記載する．

## 第 2 章

### SNS について

## 2.1 本章の構成

本章では、研究の対象である SNS(Social Networking Space)とそれに関連する解説を記す。

## 2.2 SNS とは

インターネット上の会員制サービス的一种。友人、知人間のコミュニケーションを円滑にする手段や、新たな人間関係を構築するための場を提供している。

この SNS の中では必然的にネットワークが生まれ、そのネットワークの中にコミュニティが誕生していると考ええる。

## 2.3 ネットワークとは

節点（ノード）と経路（リンクまたはエッジ）の集合からなるもの。ノード A, B, C, D, E とそれらをつなぐリンクからなるネットワーク図を以下に図 2-1 ネットワークに記す。

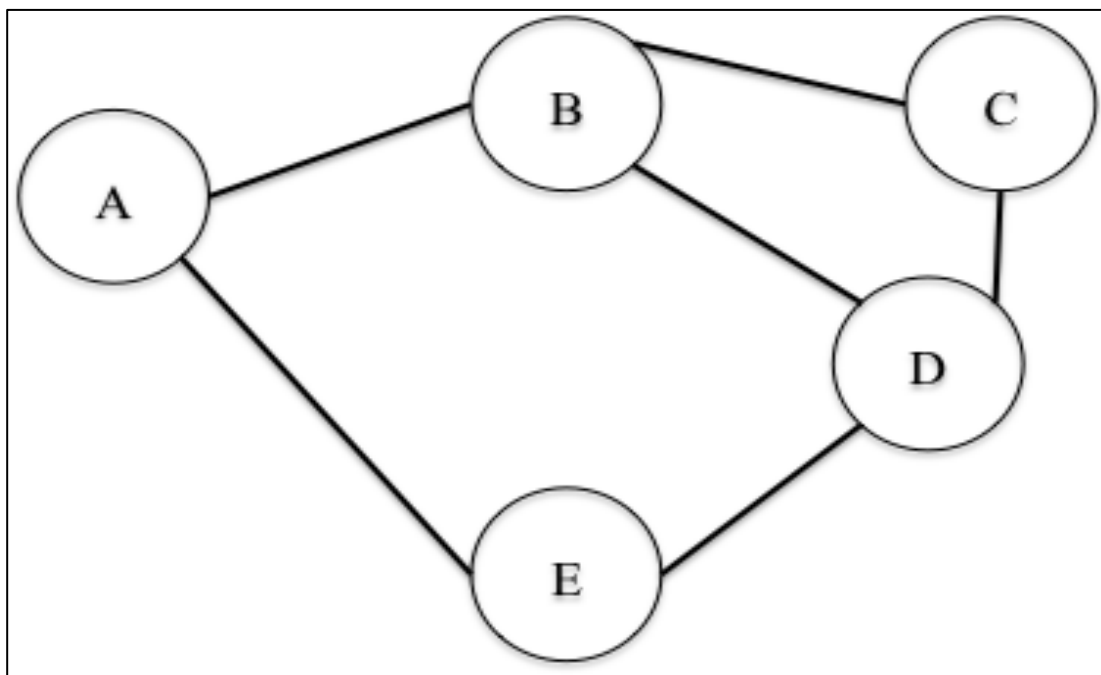


図 2-1 ネットワーク



ネットワークには大きく分けて4種類に分類することができる。[1]

1. 友人，会社，あるいは趣味の仲間同士で形成される人々のネットワーク，SNS を介したネットワークはこれに該当する。
2. WWW（World Wide Web）に代表される情報を対象物として連結された情報のネットワーク。
3. インターネット回線等のインフラ的な技術的なネットワーク。
4. 食物連鎖といった動物種や生物種の生物学的なネットワーク。

これら4種類のネットワークにはノードとリンクが必ず存在しているが，4種類のネットワークには違った流れ（フロー）が存在しており，それらは流れによって種類分けされている。

#### 2.4 コミュニティとは

コミュニティとは大きく分けて2つある。1つは同じ地域に居住して利害を共にし，政治・経済・風俗などにおいて深く結びついている地域共同体のこと。2つめはインターネット上の集まる同じ共通点を持った人間の集まりのこと。

#### 2.5 参考文献

- [1] 増山幸一, ネットワーク理論の基礎, 明治学院大学経済学部, 学術論文, 2013

## 第 3 章

### Twitter について

### 3.1 本章の構成

本章では本研究で使用する Twitter について記す.

### 3.2 Twitter とは

Twitter はツイートと呼ばれる短い文字列を投稿するためのサービスである. このツイートとは最大文字数が 140 字に制限されており, 短文での投稿しかできない. そのためユーザーからはつぶやきと呼ばれ, 鳥の鳴き声の意味であるツイートと呼ばれている.

Twitter 上で自分以外のユーザーのツイートを読むためには, そのユーザーのページにアクセスするか, そのユーザーをフォローする必要がある. フォローしているユーザーのツイートは, ひとまとめにされ, タイムラインを形成する. 誰が誰をフォローしているかという情報 (フォロー関係) は, Twitter におけるユーザーのつながりの一つの表現である.



図 3-1 Twitter 使用例

### 3.3 用語

Twitter で使用する用語に関して解説する. [1]

#### 3.3.1 ユーザー

Twitter を利用する者をユーザーと呼ぶ.

#### 3.3.2 ユーザー名(スクリーン名)

ユーザーがアカウントを登録する際にユーザー同士を区別するために設定するアカウント名. スクリーン名は最大 15 文字の半角英数字, アンダースコアで構成されている. メンションを送る際に@以下の文字として使用される.

#### 3.3.3 TwitterID

ユーザーがアカウントを登録する際に Twitter がユーザーを識別するために決定する数字の羅列である. スクリーン名は変更可能なのでシステム上では TwitterID を使用してユーザーの識別を行う.

#### 3.3.4 ツイート

Twitter における「つぶやき」を表す. 1 つのツイートには 160 文字の制限があるが, 160 文字に 20 文字の TwitterID を含むため 140 文字でツイートすることができる.

#### 3.3.5 タイムライン

フォローしているユーザーのツイートを, リアルタイムに時系列ごとに表示される一覧である.

#### 3.3.6 フォロー

特定のユーザーのツイートをタイムラインに表示するために登録すること.

### 3.3.7 フォロワー

特定のユーザーをフォローしているユーザーのことを指す。ユーザーA がユーザーB をフォローしている場合、ユーザーB はユーザーA のフォロワーである。

### 3.3.8 フレンド

特定のユーザーがフォローしているユーザーのことを指す。ユーザーA がユーザーB をフォローしている場合、ユーザーA にとってユーザーB はフレンドである。

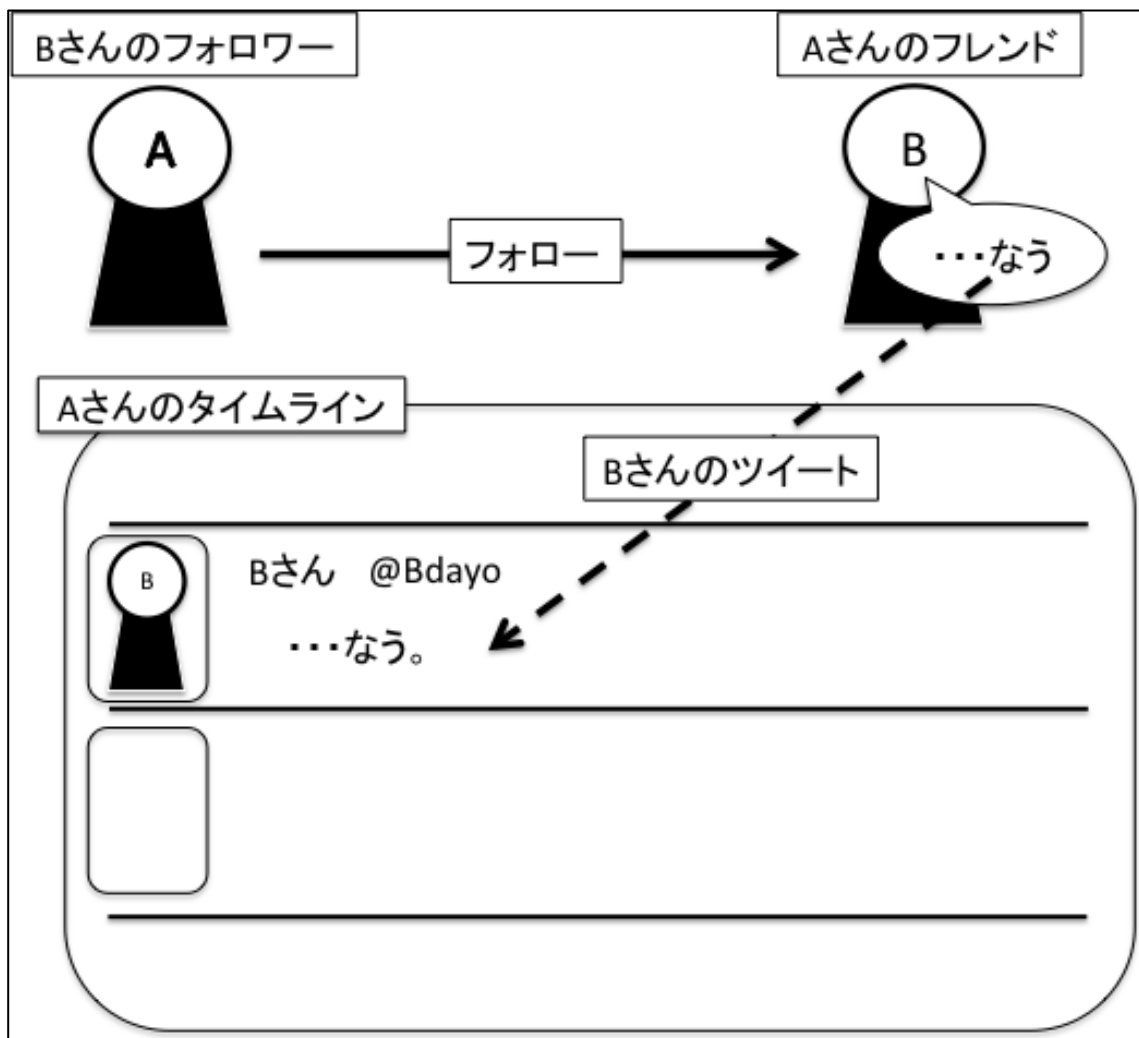


図 2-2 ツイート，フォロー，フォロワー，フレンド，タイムラインの関係

### 3.3.9 メンション (@関連ツイート)

フォローしているユーザーに宛ててツイートをする操作のこと。ツイート内に「@ユーザー名」を付けてツイートすることで、@以下のユーザーのタイムラインに表示することができる。

メンションによるツイートとタイムライン上の表示の仕組みを以下の図 3-3 に記す。ユーザーB からユーザーA へ宛てたツイート(@以下に A のユーザー名を含んだツイート)は、ユーザーA のタイムラインに表示される。それと同時にユーザーA とユーザーB をフォローしている、ユーザーD のタイムラインには、ユーザーB のツイートが表示される。しかしユーザーA のみをフォローしている、ユーザーC のタイムラインには表示されない。

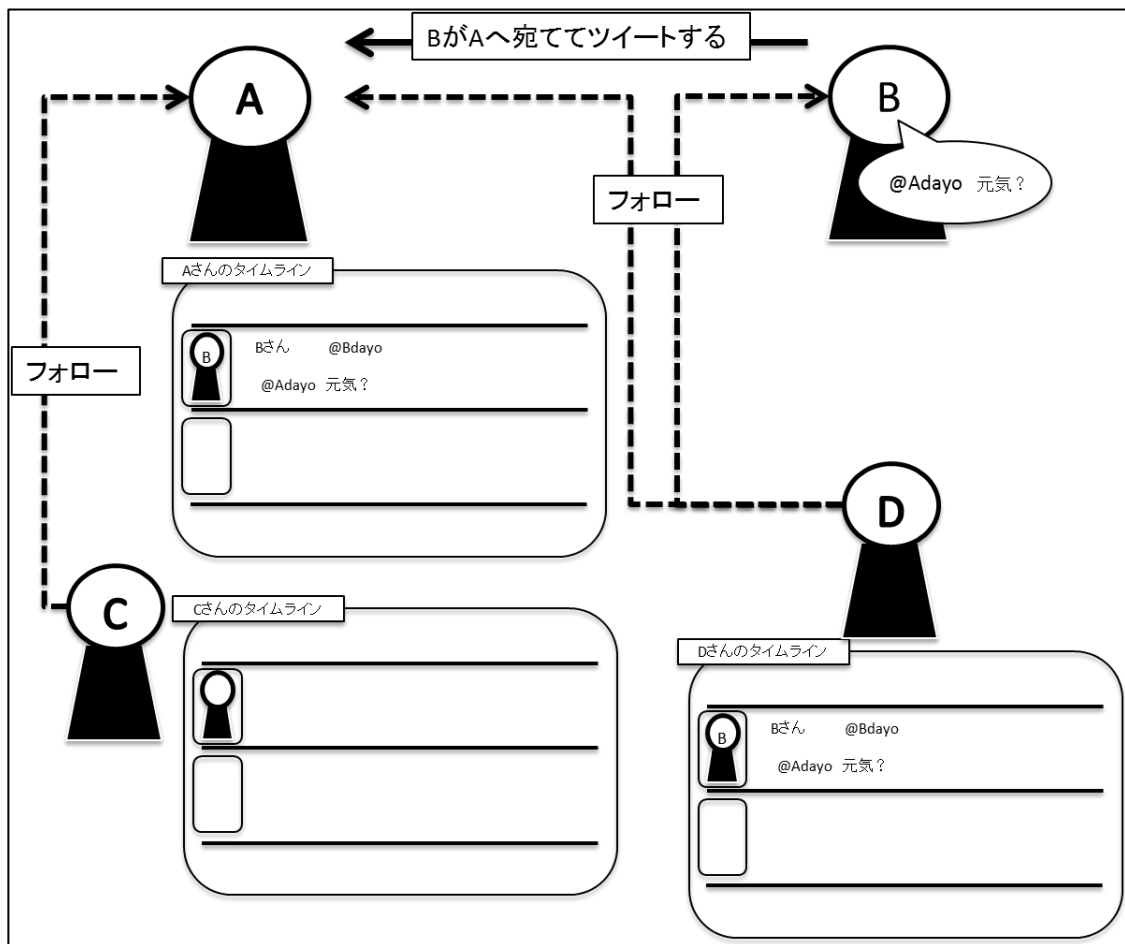


図 3-3 メンションによるツイートとタイムライン上の表示の仕組み

### 3.3.10 リツイート

他のユーザーのツイートを、自分のフォロワーに向けてツイートする操作のこと。本研究では主にこのリツイートを調べることで Twitter ユーザー同士のネットワークの形を見つけ出すことを目指した。

なお今日ではリツイートが Twitter の代表的な機能であるが、もとは Twitter の標準的な機能ではなかった。リツイートは TwitterAPI1.0 公開後に作成されたサードパーティのアプリケーションが公開していた機能である。

リツイートによるタイムラインの表示の仕組みを以下の図 3-4 に記す。ユーザー A が自分のタイムラインにある、ユーザー C のツイートをリツイートする。そうするとユーザー C をフォローしていないユーザー B のタイムラインにはユーザー A をフォローしているために、ユーザー C のツイートがユーザー B のタイムラインに表示される。

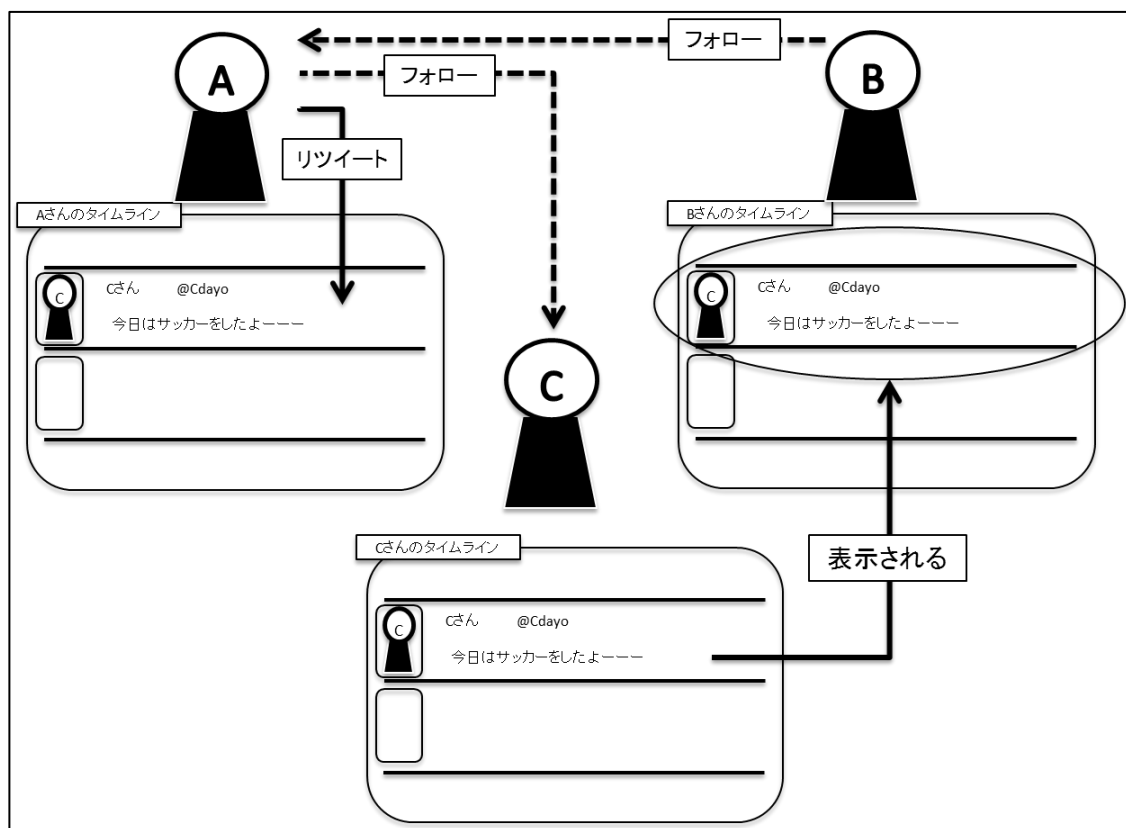


図 3-4 リツイートによるタイムラインの表示の仕組み

#### 3.3.11 お気に入り

他のユーザーのツイートを，自分のお気に入りリストへ登録する操作のこと．お気に入りを登録したユーザー以外のユーザーも，お気に入りした登録したユーザーをフォローしていれば確認することができる．

#### 3.3.12 リスト

フレンドとは別にユーザーをグループ化し，まとめて管理する機能のこと．なおリストは 500 個まで作成でき，各リストには 5000 個のアカウントを登録することができる．

#### 3.3.13 ハッシュタグ

ツイート内に記載された，#から始まる文字列のこと．ハッシュタグはカテゴライズされ，特定のハッシュタグを検索するとそれに関わるツイートが時系列に一覧表示される．

#### 3.3.14 トレンド

Twitter 上のツイートの中で，多く話題に上っているキーワードを，リアルタイムに抽出し表示する機能のこと．

#### 3.3.15 非公開ツイート

フォロワーにのみツイートを公開する設定．非公開ツイートを設定しているユーザーのツイート閲覧するためには，その相手に対してフォローリクエストを送信し許可を得ることが必要となる．



### 3.4 参考文献

- [1] 晒谷亮輔, Twitter 上の人間関係ネットワークの抽出とその分析, 千葉大学都市環境システム学科, 2011
- [2] ビズ・ストーン(2014)『ツイッターで学んだいちばん大切なこと——共同創業者の「つぶやき」』 石垣賀子訳, 早川書房
- [3] ニック・ビルトン(2014)『ツイッター創業物語 金と権力, 友情 そして裏切り』伏見 威蕃訳, 日本経済新聞出版

## 第 4 章

### ネットワーク抽出について

#### 4.1 本章の構成

本節では、本研究においてツールを作成する際に TwitterAPI と MySQL を使用する．そのため API, TwitterAPI, MySQL について調査をする．

#### 4.2 API とは

API (Application Programing Interface) はソフトウェア間で情報のやり取りをするための命令や規約を定めたインターフェイスの仕様である．API を使用することで OS (Operating System)やアプリケーションソフト, WEB アプリケーションが持つ機能の一部を外部アプリケーションから簡単に利用できるようになる．[1]

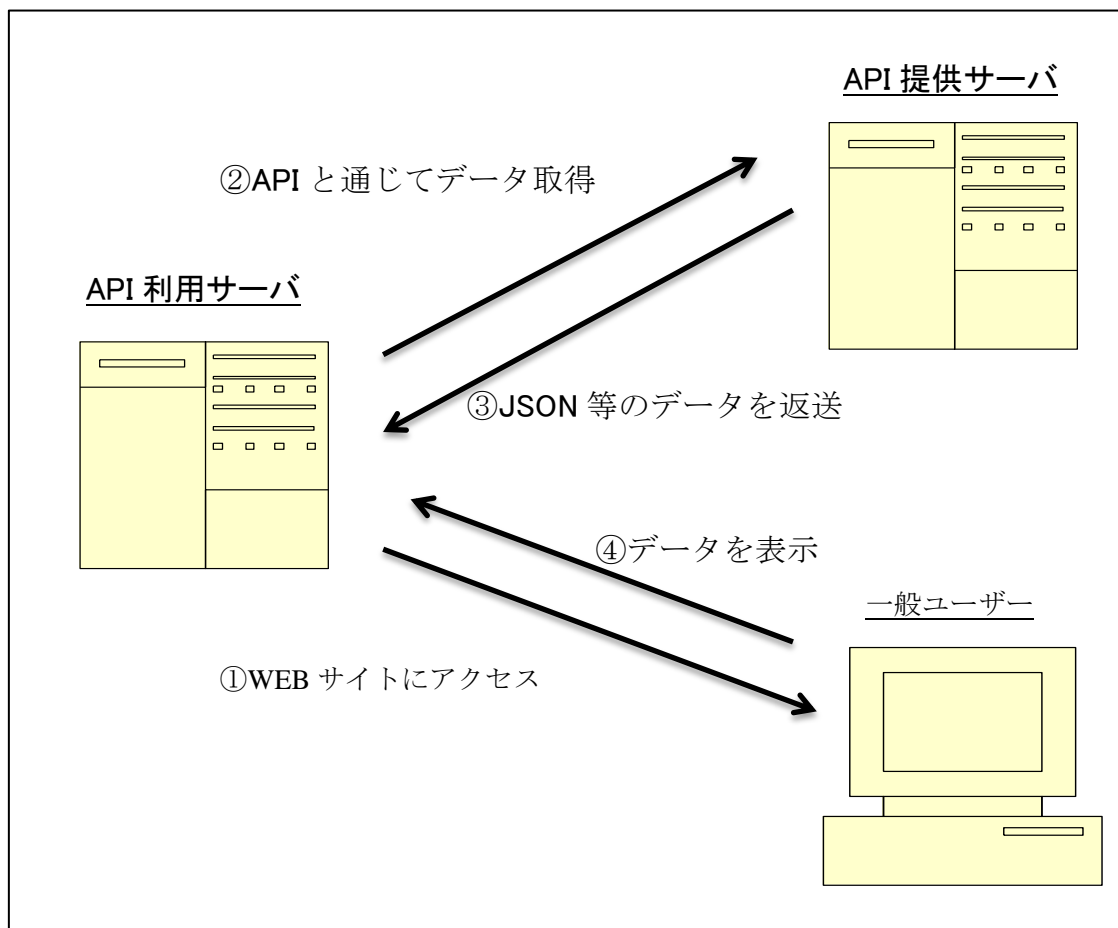
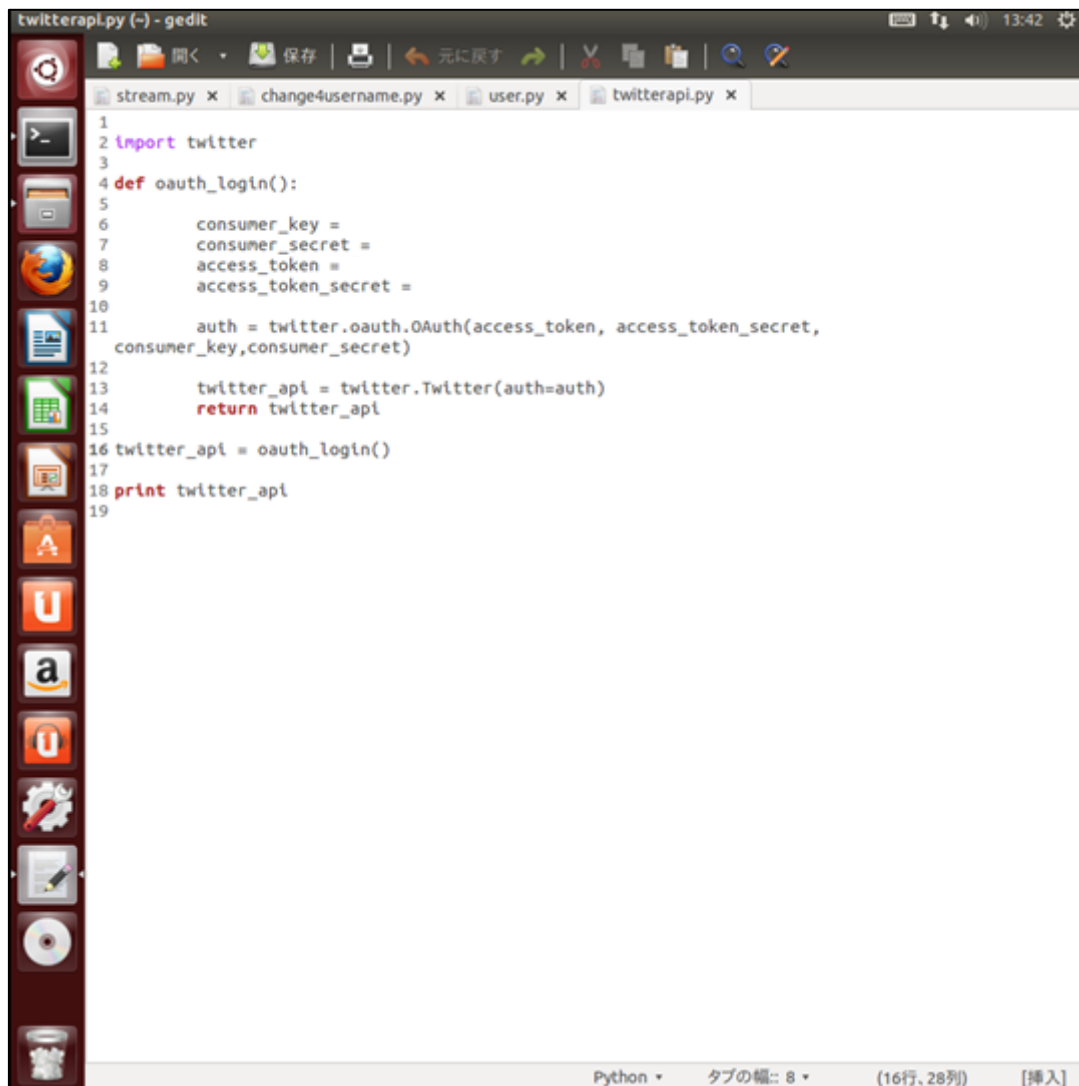


図 4-1 API イメージ図

### 4.3 TwitterAPI とは

TwitterAPI とは Twitter 社の提供している API で、Twitter のサービスを使用するために用意された。TwitterAPI には多くの種類の API を用意しているが、本研究では REST API と Streaming API の 2 種類を用いてネットワークの抽出を行う。[2]

以下に使用例の図を示す。



```
twitterapi.py (-) - gedit
1
2 import twitter
3
4 def oauth_login():
5
6     consumer_key =
7     consumer_secret =
8     access_token =
9     access_token_secret =
10
11     auth = twitter.oauth.OAuth(access_token, access_token_secret,
12                                consumer_key, consumer_secret)
13
14     twitter_api = twitter.Twitter(auth=auth)
15     return twitter_api
16
17 twitter_api = oauth_login()
18 print twitter_api
19
```

図 4-2 TwitterAPI 使用例

図の使用例のようにプログラムを実行することによって、Twitter 内のデータを JSON 形式で呼び出すことができる。

#### 4.3.1 TwitterAPI の使用方法

TwitterAPI を使う方法はさまざまにあるが、本稿では、Linux の Ubuntu で Python を使って説明をする。操作はターミナル（端末）を使用し、コマンド入力を行う。なお本節では TwitterAPI を使用する際に必要となる AccessToken の調査も行う。

#### 4.3.2 REST API とは

Twitter API のパラメータ（リソース）を指定し、特定の URL に HTTP でアクセスすると、JSON で記述されたメッセージでレスポンスされるシステムのこと。この REST API はツイートの更新や参照を行う際に使用する最も基本的な API のこと。

#### 4.3.3 REST API リスト

本項では、TwitterAPI の REST API をリスト形式に説明する。リソース内の API の種類は GET が取得を意味し、POST が投稿を意味する続くディレクトリが TwitterAPI のコードを示す。[3]

Timelines		
リソース	説明	リミット
GET statuses/home_timeline	自分のホームタイムラインツイート一覧を取得	15
GET statuses/mentions_timeline	自分のメンション付きツイート一覧を取得	15
GET statuses/retweets_of_me	リツイートされた自分のツイート一覧を取得	15
GET statuses/user_timeline	指定ユーザーのツイート一覧を取得	180

Tweets		
リソース	説明	リミット
GET statuses/show/:id	指定のツイートを取得	180
GET statuses/retweets/:id	指定ツイートのリツイート一覧を取得	15
GET statuses/oembed	指定ツイートの埋め込み用のタグを取得	180
POST statuses/update	ツイートを投稿	-
POST statuses/destroy/:id	指定のツイートを削除	-
POST statuses/retweet/:id	指定ツイートをリツイート	-
POST statuses/update_with_media	画像付きのツイートを投稿	-

Search		
リソース	説明	リミット
GET search/tweets	指定の問い合わせにマッチするツイート一覧を取得	180

Direct Message		
リソース	説明	リミット
GET direct_messages	受信したダイレクトメッセージ一覧を取得	15
GET direct_messages/sent	送信したダイレクトメッセージ一覧を取得	15
GET direct_messages/show	指定のダイレクトメッセージを取得	15
POST direct_messages/destroy	指定のダイレクトメッセージを削除	-
POST direct_messages/new	ダイレクトメッセージを送信	-

Friends, Followers		
リソース	説明	リミット
GET friends/list	自分がフォローしているユーザー一覧を取得	15
GET friends/ids	指定ユーザーがフォローしているユーザーID 一覧を取得	15
GET followers/list	自分のフォロワー一覧を取得	15
GET followers/ids	指定ユーザーをフォローしているユーザーID 一覧を取得	15
GET friendships/show	指定のユーザー同士のフォロー関係を取得	180
GET friendships/lookup	自分と指定ユーザーのフォロー関係を取得	15
GET friendships/incoming	自分にフォローリクエストしているユーザーID 一覧を取得	15
GET friendships/outgoing	自分が送信したフォローリクエストの認証が保留中のユーザーID 一覧を取得	15
GET friendships/no_retweets/ids	リツイートを表示しないように設定しているユーザー一覧を取得	15
POST friendships/create	指定のユーザーをフォロー	-
POST friendships/destroy	指定のユーザーをアンフォロー	-
POST friendships/update	指定ユーザーのデバイス通知, リツイート通知を有効/無効を変更	-

Users		
リソース	説明	リミット
GET users/show	指定のユーザーを取得	180
GET users/lookup	指定のユーザー一覧を取得(複数指定可)	180
GET users/search	指定の問い合わせにマッチするユーザー情報を取得	180
GET users/contributees	指定ユーザーの運営アカウント(ライター先)を取得	15
GET users/contributors	指定ユーザーのライターを取得	15
GET users/profile_banner	指定ユーザーのプロフィールバナーを取得	180
GET account/settings	自分の設定(トレンドの場所, 位置, 睡眠時間情報等)を取得	15
GET account/verify_credentials	自分が有効なユーザーかを取得	15
POST account/settings	自分の設定(トレンドの場所, 位置, 睡眠時間情報等)を更新	-
POST account/update_delivery_device	SMS 機能の有効/無効(日本では利用不可)	-
POST account/update_profile	自分のプロフィールを変更	-
POST account/update_profile_background_image	自分のプロフィール背景を変更	-
POST account/update_profile_colors	自分のプロフィール画面の各種色を変更	-
POST account/update_profile_image	自分のプロフィール画像を変更	-
POST account/remove_profile_banner	自分のプロフィールバナーを削除	-
POST account/update_profile_banner	自分のプロフィールバナーを変更	-
GET blocks/list	自分がブロックしているユーザー一覧を取得	15
GET blocks/ids	自分がブロックしているユーザー ID を取得	15



POST blocks/create	指定のユーザーをブロック	-
POST blocks/destroy	指定のユーザーのブロックを解除	-

Suggested Users		
リソース	説明	リミット
GET users/suggestions/:slug	指定カテゴリのおすすめユーザー一覧を取得	15
GET users/suggestions	おすすめユーザーのカテゴリリストを取得	15
GET users/suggestions/:slug/members	指定カテゴリのおすすめユーザーのツイート一覧を取得	15

Favorites		
リソース	説明	リミット
GET favorites/list	指定ユーザーのお気に入り一覧を取得	15
POST favorites/destroy	指定ツイートのお気に入りを取消	-
POST favorites/create	指定ツイートをお気に入り	-

Lists		
リソース	説明	リミット
GET lists/list	指定ユーザーのリスト一覧を取得	15
GET lists/ownerships	指定ユーザーのリスト一覧を取得. 自分を指定した場合はプライベートリストのみ取得	15
GET lists/show	指定のリストを取得	15
GET lists/statuses	指定リストのツイート一覧を取得	180
GET lists/subscriptions	指定ユーザーがフォローしているリスト一覧を取得	15
GET lists/memberships	指定ユーザーがフォローされているリスト一覧を取得	15
POST lists/destroy	指定のリストを削除	-
POST lists/update	指定のリスト情報の設定(リスト名, パブリック/プライベート, 説明)を変更	-
POST lists/create	リストを作成	-
GET lists/subscribers	指定リストをフォローしているユーザー一覧を取得	180
GET lists/subscribers/show	指定ユーザーが指定リストをフォローしているかを取得	15
POST lists/subscribers/create	指定のリストをフォロー	-
POST lists/subscribers/destroy	指定のリストをアンフォロー	-
GET lists/members/show	自分の指定リストに指定のユーザーが追加されているかを取得	15
GET lists/members	自分の指定リストに追加しているユーザー一覧を取得	180
POST lists/members/create	自分の指定リストに指定ユーザーを追加	-
POST lists/members/create_all	自分の指定リストに指定ユーザーを追加(複数指定可)	-
POST lists/members/destroy	自分の指定リストから指定ユーザーを削除	-

POST lists/members/destroy_all	自分の指定リストから指定ユーザーを削除(複数指定可)	-
-----------------------------------	----------------------------	---

Saved Searches		
リソース	説明	リミット
GET saved_searches/list	検索メモ一覧を取得	15
GET saved_searches/show/:id	検索メモを取得	15
POST saved_searches/create	検索メモを作成	-
POST saved_searches/destroy/:id	検索メモを削除	-

Place, Geo code		
リソース	説明	リミット
GET geo/id/:place_id	指定ジオコードから場所情報を取得	15
GET geo/reverse_geocode	指定の緯度・経度から場所情報を取得	15
GET geo/search	指定の情報から場所情報を取得	15
GET geo/similar_places	指定の緯度・経度、地名から場所情報一覧を取得	15
POST geo/place	指定の情報から場所情報を作成	-

Trends		
リソース	説明	リミット
GET trends/place	指定 WOEID のトレンドを取得	15
GET trends/available	有効なトレンド地域一覧を取得	15
GET trends/closest	指定の緯度・経度から最も近いトレンド地域を取得	15

Spam Reporting		
リソース	説明	リミット
POST users/report_spam	指定ユーザーをスパム報告	-

OAuth		
リソース	説明	リミット
GET oauth/authenticate	ウェブブラウザ用の OAuth ユーザー認証	-
GET oauth/authorize	OAuth ユーザー認証	-
POST oauth/access_token	アクセストークンを取得	-
POST oauth/request_token	リクエストトークンを取得	-
POST oauth2/token	OAuth 2 ユーザー認証	-
POST oauth2/invalidate_token	OAuth 2 Bearer Token を無効にする	-

Help		
リソース	説明	リミット
GET help/configuration	Twitter の各種	15
GET help/languages	Twitter でサポートしている言語一覧を取得	15
GET help/privacy	Twitter のプライバシーポリシーを取得	15
GET help/tos	Twitter のサービス利用規約を取得	15
GET application/rate_limit_status	API のリクエスト残数/リセット時間を取得	180

#### 4.3.4 Streaming API とは

Streaming API は比較的新しい API で、タイムラインの変更をリアルタイムに 受け取れるもの。Streaming API も REST を使用して Twitter API へアクセスするが、タイムラインが更新されるまでレスポンスを返すのを保留させるので、リアルタイム性を確保している。

Streaming		
リソース	説明	リミット
POST statuses/filter	指定の問い合わせにマッチするツイートをストリーミングで取得	-
GET statuses/sample	全ツイートからランダムに選ばれたサンプルツイートをストリーミングで取得	-
GET statuses/firehose	全ツイートをストリーミングで取得(特別な許可が必要)	-
GET user	自分のタイムライン, イベントをストリーミングで取得	-
GET site	複数ユーザーのタイムライン, イベントをストリーミングで取得	-

#### 4.3.5 レートリミットとは

レートリミットとは、各 API の呼び出し制限数と制限がリセットされる時間のこと。各 API は制限数を使い切ると使用できなくなるが、リセット時間が始めの呼び出しからカウントされ、そのリセット時間が経過すると制限数は元に戻る。TwitterAPI 1.1 での制限時間は 15 分と設定されている。

#### 4.3.6 AccessToken とは

AccessToken とはアプリケーションが、ユーザーの代わりに、Twitter へアクセスするために必要なユーザー情報を暗号化したものである。OAuth 認証を行うことで、AccessToken を取得する。OAuth 認証の方法は以下の図 4-3 OAuth 認証の方法に示す。[4]

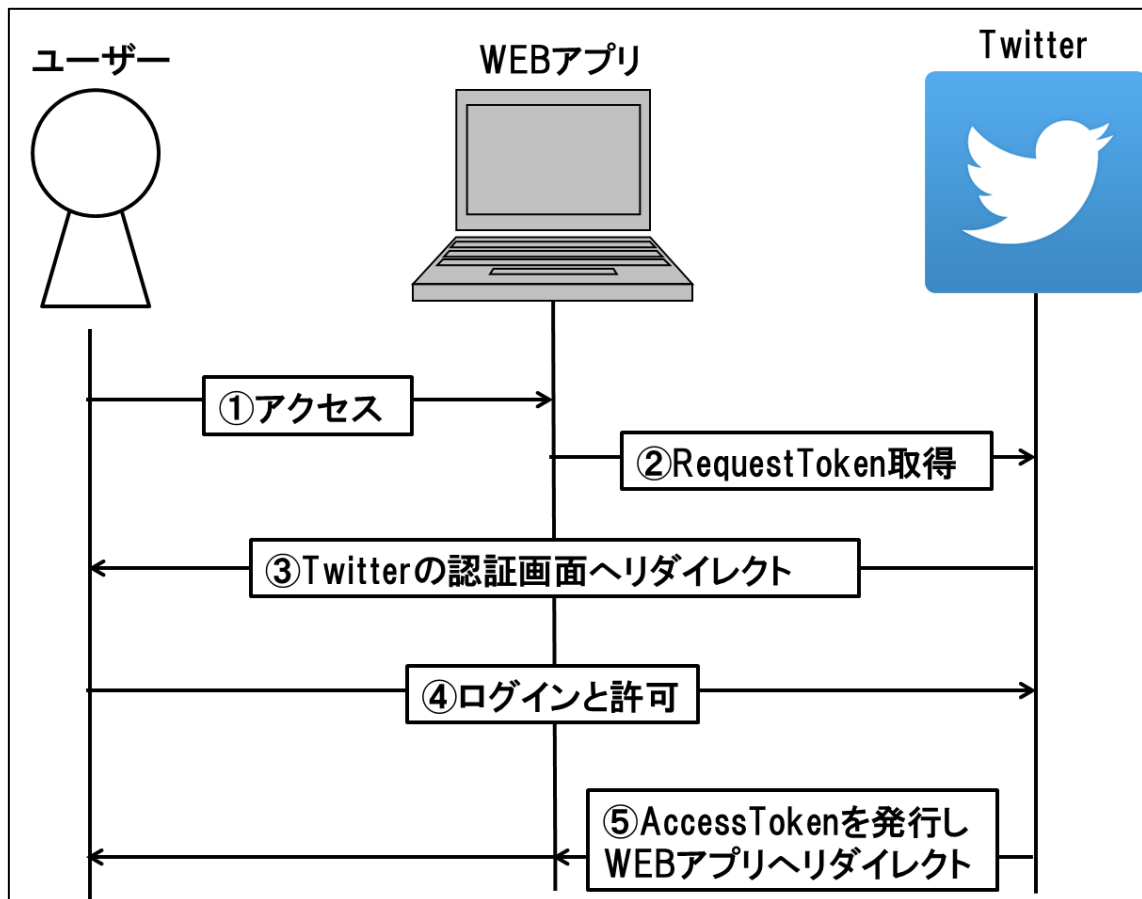


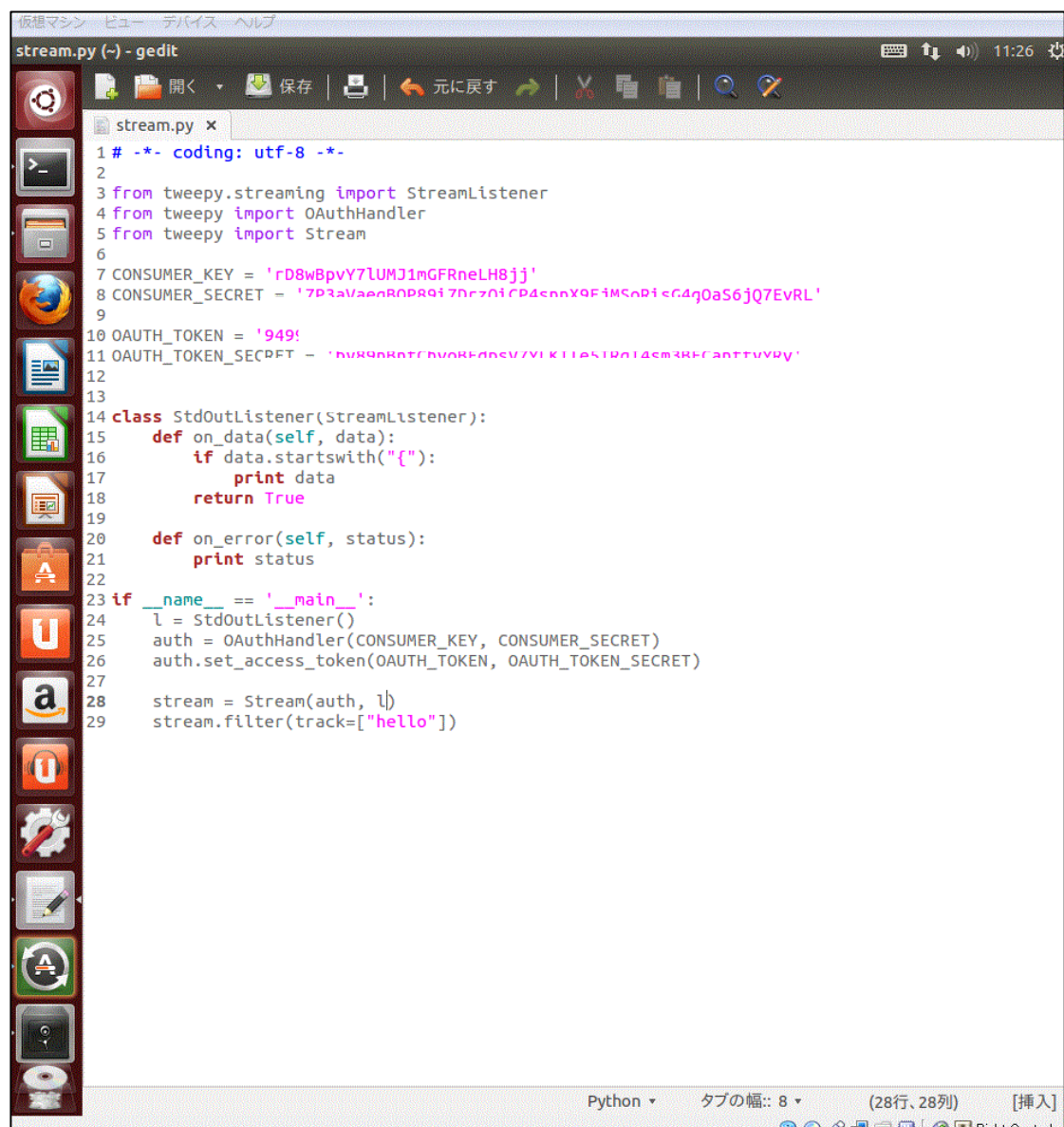
図 4-3 OAuth 認証の方法

#### 4.4 TwitterAPI を実際に使ってみる

TwitterAPI を使用する方法は様々な方法があるが、本稿では Linux の Ubuntu の端末上で言語は Python を使って説明する。

StreamingAPI を使用し、「hello」と付くツイートをリアルタイムで出力した際の例を以下の

図 4-4 Streaming API 使用例に記す。



```
stream.py (-) - gedit
1 # -*- coding: utf-8 -*-
2
3 from tweepy.streaming import StreamListener
4 from tweepy import OAuthHandler
5 from tweepy import Stream
6
7 CONSUMER_KEY = 'rD8wBpY7LUMJ1mGFRneLH8jj'
8 CONSUMER_SECRET = '7P3aVaaRnD89i7Hr7niCP4ennX9F4MSnPi6C4q0a56jQ7EvRL'
9
10 OAUTH_TOKEN = '949!'
11 OAUTH_TOKEN_SECRET = 'hV89nRntfDVORfDnSVYK11e5120145m3REfANTVvPY'
12
13
14 class StdOutListener(StreamListener):
15     def on_data(self, data):
16         if data.startswith("{"):
17             print data
18             return True
19
20     def on_error(self, status):
21         print status
22
23 if __name__ == '__main__':
24     l = StdOutListener()
25     auth = OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
26     auth.set_access_token(OAUTH_TOKEN, OAUTH_TOKEN_SECRET)
27
28     stream = Stream(auth, l)
29     stream.filter(track=["hello"])
```

図 4-4 Streaming API 使用例



図 4-4 使用例の Ubuntu の端末上に JSON 形式で表示された結果を以下の図 4-5 使用例結果に記す。

```
{
  "created_at": "Fri Oct 17 02:26:15 +0000 2014",
  "id": 522936599240540160,
  "id_str": "522936599240540160",
  "text": "@jenreid26 @ItsMrsMoran @bibliobibuliboo @Zeusy4lif e @LRWagner1 @labrys75 @_melsroom @jobean_1964 @DyanSohn @_luckycharmz_ @suz2311 \nHello",
  "source": "\u003c a href=\"http://twitter.com/download/android\" rel =\"nofollow\" \u003eTwitter for Android\u003c/a\u003e",
  "truncated": false,
  "in_reply_to_status_id": 522930372162899968,
  "in_reply_to_status_id_str": "522930372162899968",
  "in_reply_to_user_id": 1511224255,
  "in_reply_to_user_id_str": "1511224255",
  "in_reply_to_screen_name": "jenreid26",
  "user": {
    "id": 2789141201,
    "id_str": "2789141201",
    "name": "John Craig",
    "screen_name": "jcraig3976",
    "location": "",
    "url": null,
    "description": "Just me... John Craig",
    "protected": false,
    "verified": false,
    "followers_count": 29,
    "friends_count": 83,
    "listed_count": 0,
    "favourites_count": 518,
    "statuses_count": 306,
    "created_at": "Sun Sep 28 15:20:45 +0000 2014",
    "utc_offset": null,
    "time_zone": null,
    "geo_enabled": false,
    "lang": "en",
    "contributors_enabled": false,
    "is_translator": false,
    "profile_background_color": "C0DEED",
    "profile_background_image_url": "http://abs.twimg.com/images/themes/theme1/bg.png",
    "profile_background_image_url_https": "https://abs.twimg.com/images/themes/theme1/bg.png",
    "profile_background_tile": false,
    "profile_link_color": "0084B4",
    "profile_sidebar_border_color": "C0DEED",
    "profile_sidebar_fill_color": "DDEEFF",
    "profile_text_color": "333333",
    "profile_use_background_image": true,
    "profile_image_url": "http://pbs.twimg.com/profile_images/522334992396201985/DnNCNi_r_normal.jpeg",
    "profile_image_url_https": "https://pbs.twimg.com/profile_images/522334992396201985/DnNCNi_r_normal.jpeg",
    "profile_banner_url": "https://pbs.twimg.com/profile_banners/2789141201/1411918905",
    "default_profile": true,
    "default_profile_image": false,
    "following": null,
    "follow_request_sent": null,
    "notifications": null,
    "geo": null,
    "coordinates": null,
    "place": null,
    "contributors": null,
    "retweet_count": 0,
    "favorite_count": 0,
    "entities": {
      "hashtags": [],
      "trends": [],
      "urls": [],
      "user_mentions": [
        {
          "screen_name": "jenreid26",
          "name": "Jennifer Reid",
          "id": 1511224255,
          "id_str": "1511224255",
          "indices": [0, 10]
        },
        {
          "screen_name": "ItsMrsMoran",
          "name": "Cynthia",
          "id": 822494834,
          "id_str": "822494834",
          "indices": [11, 23]
        },
        {
          "screen_name": "bibliobibuliboo",
          "name": "Melissa Hartley",
          "id": 2800796047,
          "id_str": "2800796047",
          "indices": [24, 40]
        },
        {
          "screen_name": "Zeusy4life",
          "name": "\u2728Janice\u2728",
          "id": 2164416527,
          "id_str": "2164416527",
          "indices": [41, 52]
        },
        {
          "screen_name": "LRWagner1",
          "name": "LeAnne ",
          "id": 545622547,
          "id_str": "545622547",
          "indices": [53, 63]
        },
        {
          "screen_name": "labrys75",
          "name": "Amber Liv",
          "id": 1539718398,
          "id_str": "1539718398",
          "indices": [64, 73]
        },
        {
          "screen_name": "_melsroom",
          "name": "Mel",
          "id": 127207172,
          "id_str": "127207172",
          "indices": [74, 84]
        },
        {
          "screen_name": "jobean_1964",
          "name": "Mary Jo McKay",
          "id": 399186340,
          "id_str": "399186340",
          "indices": [85, 97]
        },
        {
          "screen_name": "DyanSohn",
          "name": "Dyan Sohn",
          "id": 2241331447,
          "id_str": "2241331447",
          "indices": [98, 107]
        },
        {
          "screen_name": "_luckycharmz_",
          "name": "Tanya",
          "id": 2199608910,
          "id_str": "2199608910",
          "indices": [108, 122]
        },
        {
          "screen_name": "suz2311",
          "name": "\u2728SUZ\u2728",
          "id": 71168500,
          "id_str": "71168500",
          "indices": [123, 131]
        }
      ]
    },
    "symbols": [],
    "favorited": false,
    "retweeted": false,
    "possibly_sensitive": false,
    "filter_level": "medium",
    "lang": "en",
    "timestamp_ms": "1413512775615"
  }
}
```

図 4-5 使用例結果

#### 4.5 MySQL とは

MySQL とはオラクルが開発を行うオープンソースの RDBMS(Relational Database Management System)である。MySQL は商用利用を除いて無償で入手し、利用することができる。SQL という RDBMS を操作するための言語を使って MySQL はデータベースを作成し、その中にテーブルを作成し使用する形になっている。さらに MySQL では多くの関数や機能を使用することができるため MySQL は広く一般的に使用されている RDBMS である。本研究では MySQL 上でデータベース 1 つとテーブルを 4 つ使用して研究を行った。

MySQL の形は以下の図 4-6 MySQL の形に記す。

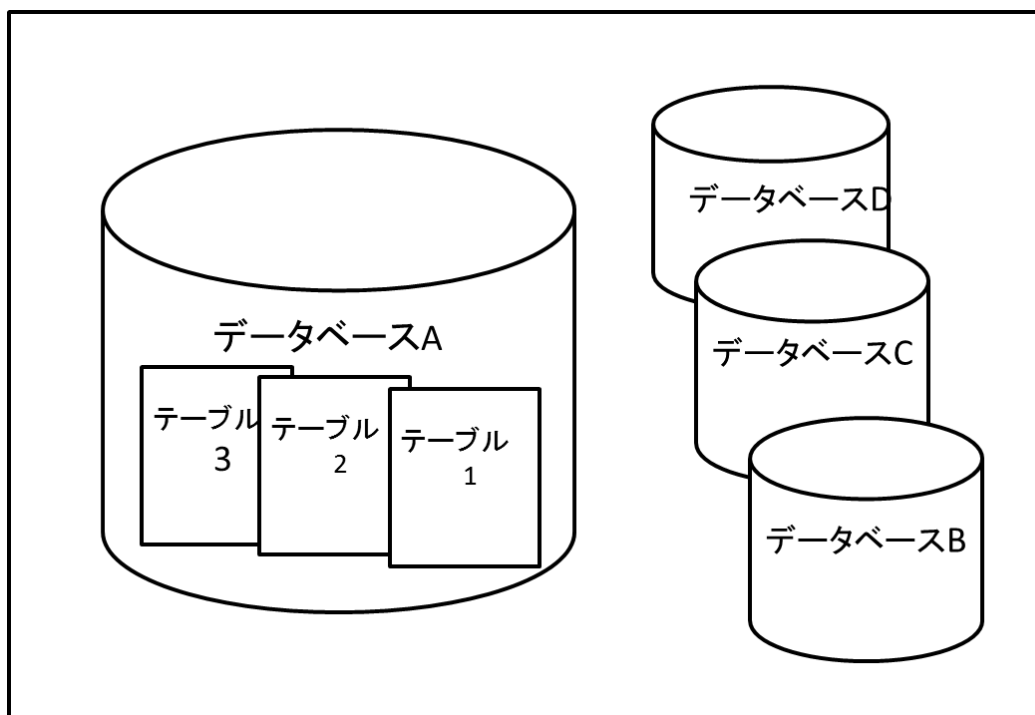


図 4-6 MySQL の形

#### 4.5.1 MySQL の導入

今回は本研究で使用した Ubuntu 内での MySQL の導入について記す。[7]

まず 5.2.1 研究を行うための用意 内に記載がある、「端末」上で以下のコマンドを使用し、インストールを行い root ユーザーに対するパスワード設定画面が表示されるので設定を行う。

```
sudo apt-get install mysql-server
```

「端末」上でユーザー名を指定し、パスワードを要求する。

```
mysql -u root -p
```

パスワードを入力することでログインできる。

#### 4.5.2 MySQL ユーザー作成

もし root 以外のユーザーを作成する場合。

root ユーザーでログインした状態で以下のコマンドを入力する。

```
mysql> GRANT ALL PRIVILEGES ON *.* TO (ユーザ名)@localhost IDENTIFIED BY '(パスワード)' WITH GRANT OPTION;  
FLUSH PRIVILEGES;
```

「端末」上でユーザー名を入力し、パスワードを要求する。

```
mysql> mysql -u (ユーザ名) -p
```

パスワードを入力しログインできるか確認する。

#### 4.5.1 MySQL を実際に試してみる

今回は Linux の端末上で SQL 文を使用して MySQL 上に上記図 4-6 にあるデータベース A、テーブル 1 を作成しテーブル 1 の削除を行うまでの手順を示す。なお Linux への MySQL の導入とユーザー名とパスワードは設定されておりログインしてある状態と仮定する。

①以下の SQL でデータベース A(databaseA)の生成を行う。

```
mysql>CREATE DATABASE databaseA;
```

②テーブル 1 を以下表にあるテーブル情報のように作成する。

表 テーブル 1 の情報

列名	データ型
A	BIT
B	BIT
C	BIT

```
mysql> CREATE TABLE table1(A BIT,B BIT,C,BIT);
```

③databaseA を使用することを宣言する。

```
mysql>USE databaseA;
```

④この状態でデータベースの中身を表示すると以下の図 4-7 のようになる。

```
mysql> SHOW tables;
```

```
+-----+
| Tables_in_databaseA |
+-----+
| table1               |
+-----+
```

図 4-7 データベース A の中身

⑤この状態でテーブル 1 の中身を表示すると以下の図 4-8 ようになり, テーブルの中身を確認することができる.

```
mysql> DESC table1;
```

Field	Type	Null	Key	Default	Extra
A	bit(1)	YES		NULL	
B	bit(1)	YES		NULL	
C	bit(1)	YES		NULL	

図 4-8 テーブル 1 の中身

⑥このテーブルにカラムを追加する.

```
mysql> INSERT INTO table1(A,B,C) VALUES(1,1,0);
```

⑥このテーブル 1 は以下のような SQL で削除することができる.

```
mysql> DROP TABLE table1;
```

#### 4.6 参考文献

- [1] 金本貴志.コンピュータの基礎知識.カルテットコミュニケーションズリスティング広告情報ブログ,<http://quartet-communications.com/info/topics/5752>(参照 2014-09-22)
- [2] Yuta Arai.TwitterAPI の使い方アクセストークンの取得方法,Syncer.<http://syncer.jp/twitter-api-how-to-get-access-token>(参照 2014-10-07)
- [3] YU SUGAWARA.【保存版】TwitterAPI1.1 REST API 全項目解説,DX.univ.<http://dx.24-7.co.jp/twitterapi1-1-rest-api/>(参照 2014-10-07)
- [4] ゼロから学,OAuth.gihyo.jp.技術評論社.<http://gihyo.jp/dev/feature/01/oauth/0001>(参照 2014-10-11)
- [5] MySQL 初心者入門講座, <http://mysqlweb.net/>,(参照 2014-01-20)
- [6] 矢吹太郎(2011)『Web アプリケーション構築入門(第2版)』森北出版株式会社

# 第 5 章

## 本論

## 5.1 本章について

本章では実際に行った研究の方法について記載した．研究の方法については研究の事前準備と研究の手順，そして実際に研究で使用したコードについて記載する．

## 5.2 研究方法

本研究の研究方法を説明する際に用語は以下の表と図5-1の本研究の研究概念図を使用する．Twitter内にあるユーザー間にあるつながりのネットワークを見ることでTwitterにあるコミュニティの抽出をすることを目指している．そのためユーザー間のつながりのネットワークを見るために，まずTwitterの機能であるリツイート調べることを考えた．そのためこの研究はリツイートを観測し，そのリツイートデータを収集することから始めた．データを収集するために使用したプログラムの形を図5-3プログラム概要図に本研究に使用したデータベース「Twitter」内の形を図5-4のER図に，テーブルの形は以下の表にテーブル定義書として記載する．

本研究では指標 1 と指標 2 を導き出す．そのために TwitterAPI の User streams の機能を使い，自分のフォローしているユーザーの行ったリツイートのみを抽出し保存する．この保存されたリツイートの中からリツイート A を使用し研究を行ったとする．この場合はユーザーB のフォローしている相手とツイート A をリツイートしたユーザーB 以外の相手を取得する．取得したリツイートの全体と指標 1, 指標 2 の関係は以下の図 5-2 のようになる．

表5-1 本研究に使用する用語

リツイート A	観測されたリツイート
ツイート A	リツイート A のオリジナルのツイート
ユーザーA	ツイート A のユーザー
ユーザーB	リツイート A のユーザー
指標 1	ユーザーB がユーザーA をフォローしていない確率
指標 2	ユーザーB がツイート A をリツイートした人たちをフォローしていない確率



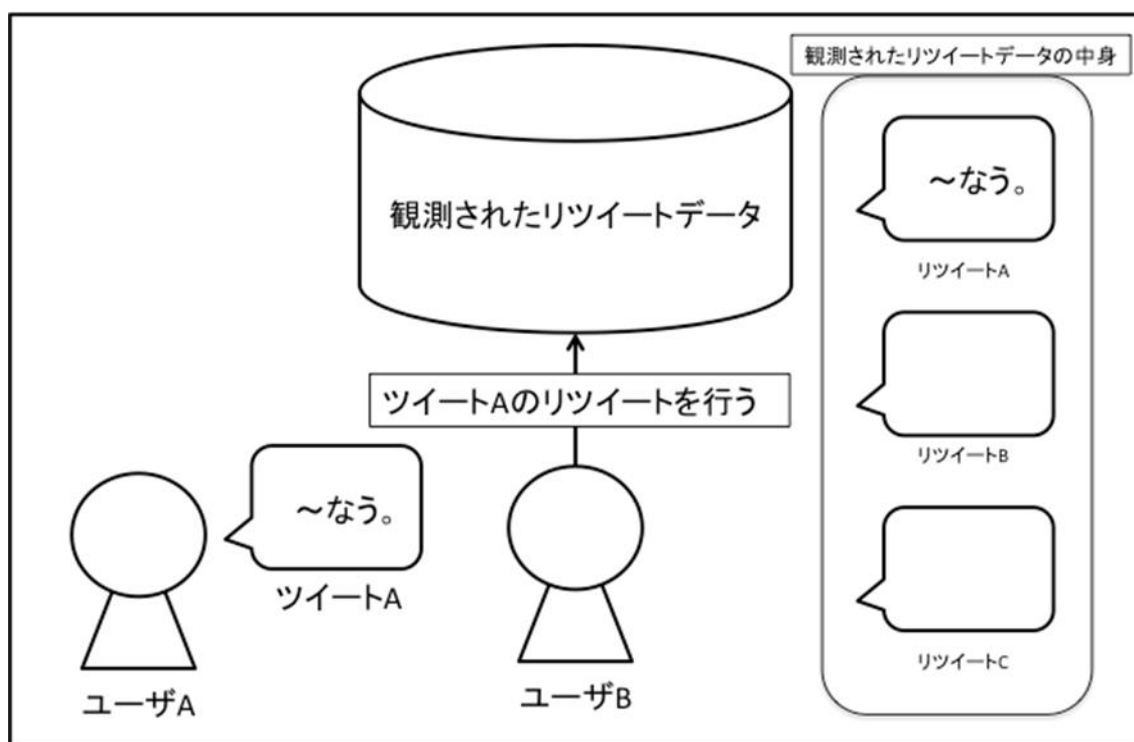


図5-1 本研究の研究概念図

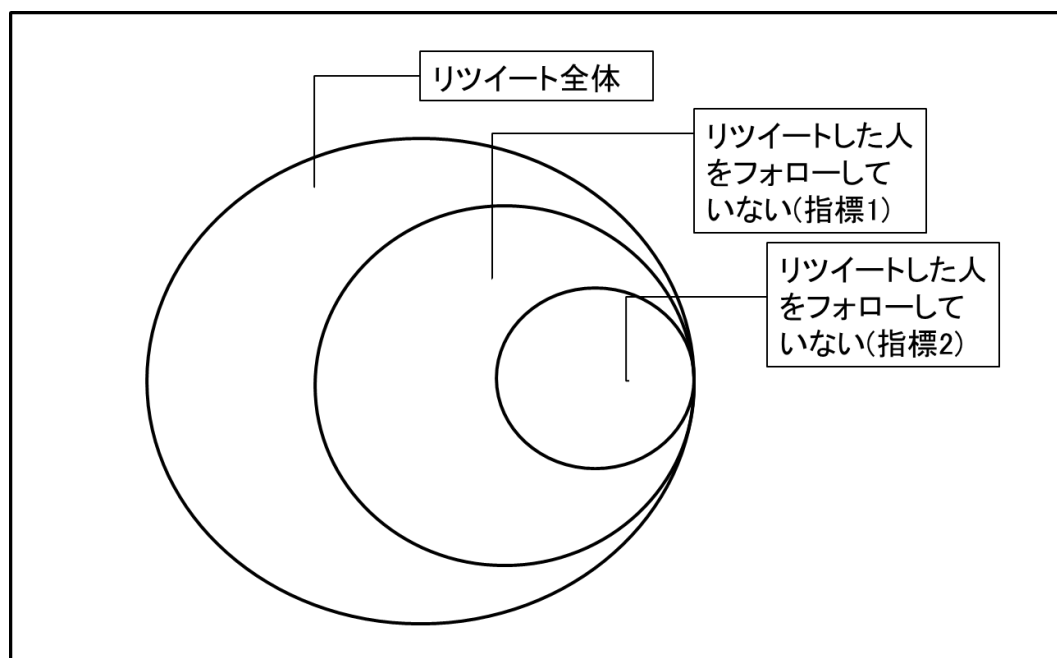


図5-2 リツイート全体，指標1，指標2の関係

表 テーブル定義書

テーブル論理名	ユーザーB情報
テーブル物理名	users

列名	データ型	長さ	NOT NULL	デフォルト	主キー	備考
id	bigint	20	YES	NULL	PRI	
screenName	varchar	100	YES	NULL	UNI	
statuses	int	11		NULL		
friends	int	11		NULL		
followers	int	11		NULL		
profoleImageUrl	varchar	1000		NULL		
rekognition	text			NULL		
friendsChecked	tinyint	1	YES	0	MUL	

テーブル論理名	リツイート情報
テーブル物理名	retweets

列名	データ型	長さ	NOT NULL	デフォルト	主キー	備考
id	bigint	20	YES	NULL	PRI	
retweet	bigint	20	YES	NULL	MUL	
retweeted	bigint	20	YES	NULL	MUL	
rcount	int	11	YES	NULL		
fcount	int	11	YES	NULL		
retweetersChecked	tinyint	1	YES	0	MUL	

テーブル論理名	リツイートしているユーザー一覧
テーブル物理名	retweeters

列名	データ型	長さ	NOT NULL	デフォルト	主キー	備考
id	int	11	YES	NULL	PRI	auto_increment
tweetID	bigint	20	YES	NULL	MUL	
retweet	bigint	20	YES	NULL	MUL	リツイートした人の ID

テーブル論理名	フォローしているユーザー一覧
テーブル物理名	friends

列名	データ型	長さ	NOT NULL	デフォルト	主キー	備考
id	int	11	YES	NULL	PRI	auto_increment
user	bigint	20	YES	NULL	MUL	
friend	bigint	20	YES	NULL	MUL	

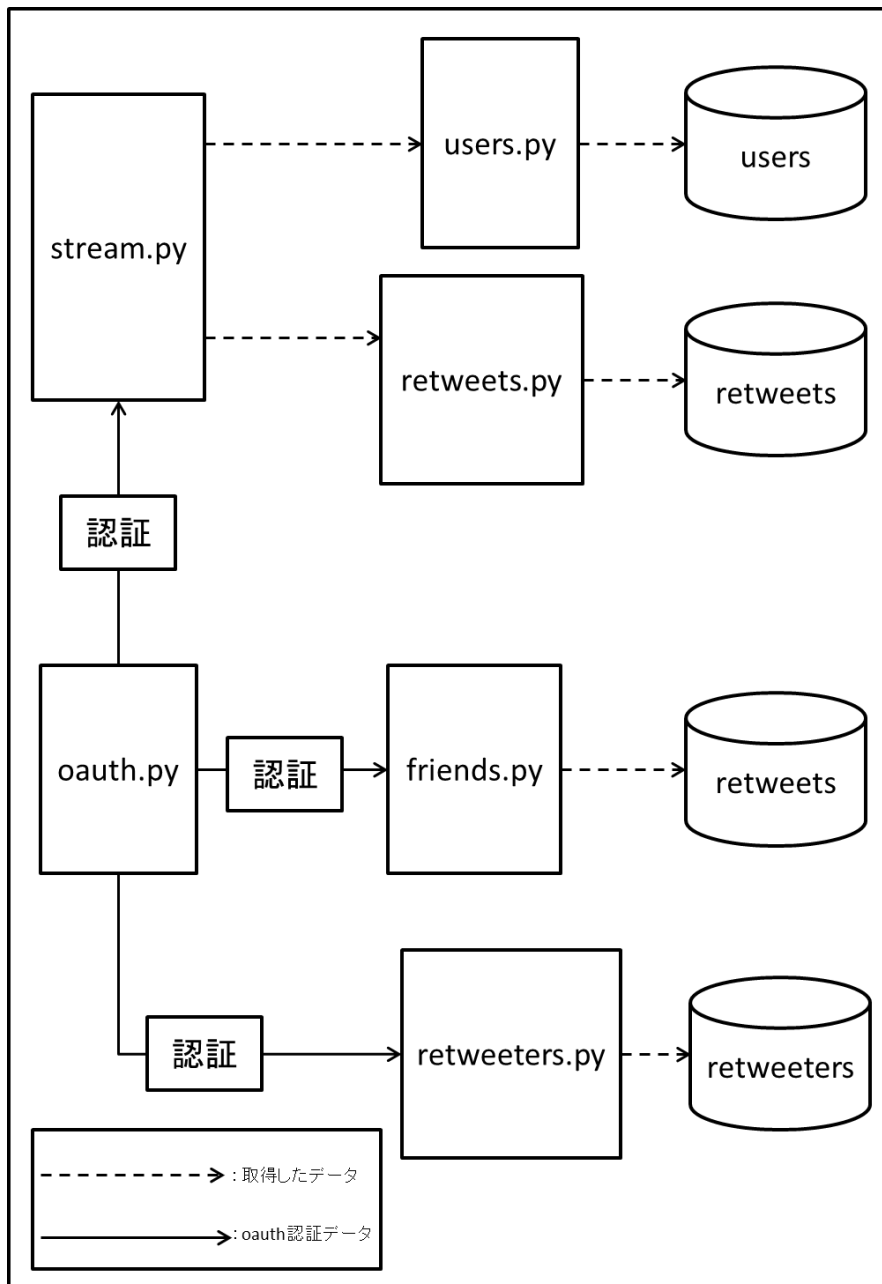


図5-3 プログラムの概要図

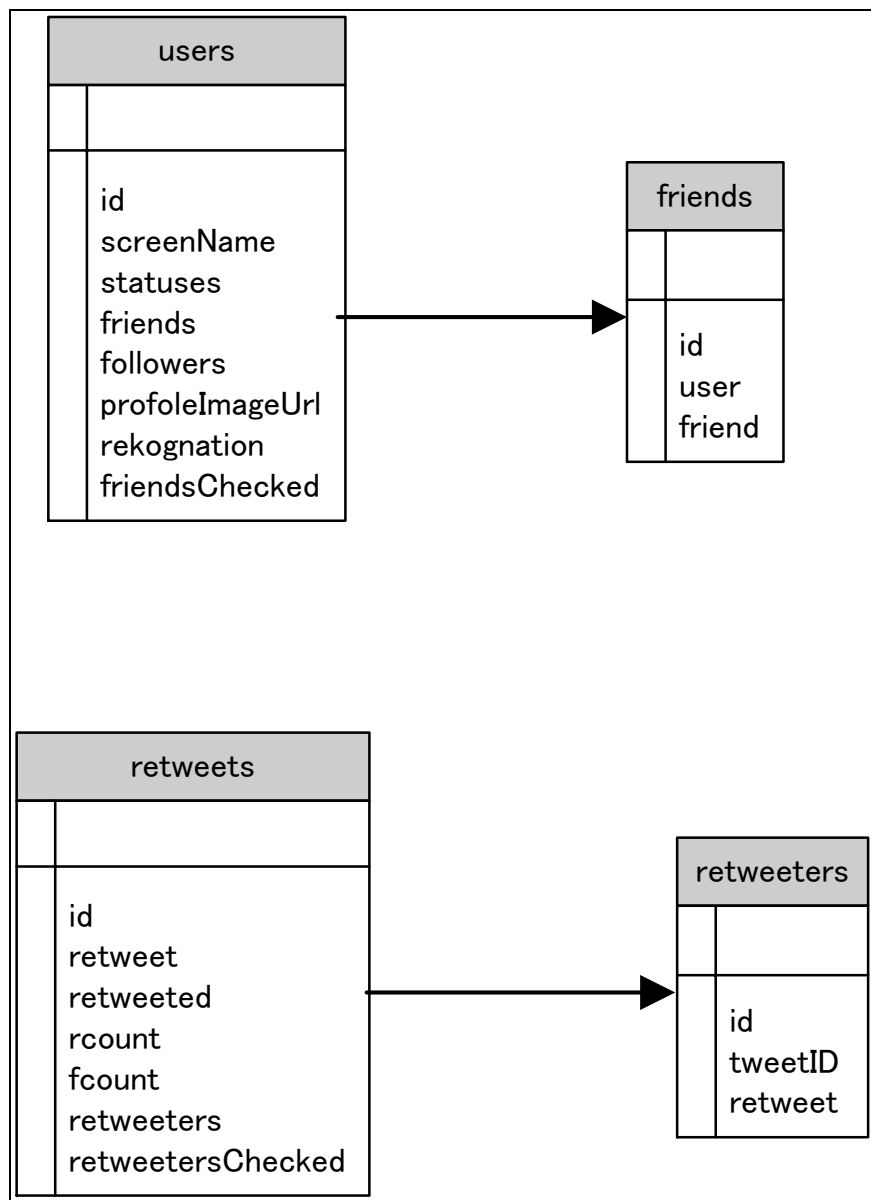


図5-4 研究に使用したデータベース「Twitter」のER図

### 5.2.1 研究を行うための用意

本研究では WindowsOS のコンピュータに Linux のディストリビューションである Ubuntu を仮想ソフト上で動かし研究を行った. 本項ではこの Ubuntu の導入方法と使用方法を記す.

[1]

#### Ubuntu の導入方法

- ① 仮想化ソフトを使用し, Ubuntu のインストールを行うための仮想マシンを作成する.
- ② Ubuntu の ISO イメージのダウンロードを行う. <http://www.ubuntulinux.jp/>へアクセスし日本語 remix の Ubuntu のダウンロードを行う.
- ③ 作成した仮想マシン上に Ubuntu のインストールを行う. インストールが完了すると以下の図 5-5 Ubuntu14.04 デスクトップ画面のような画面が表示される.

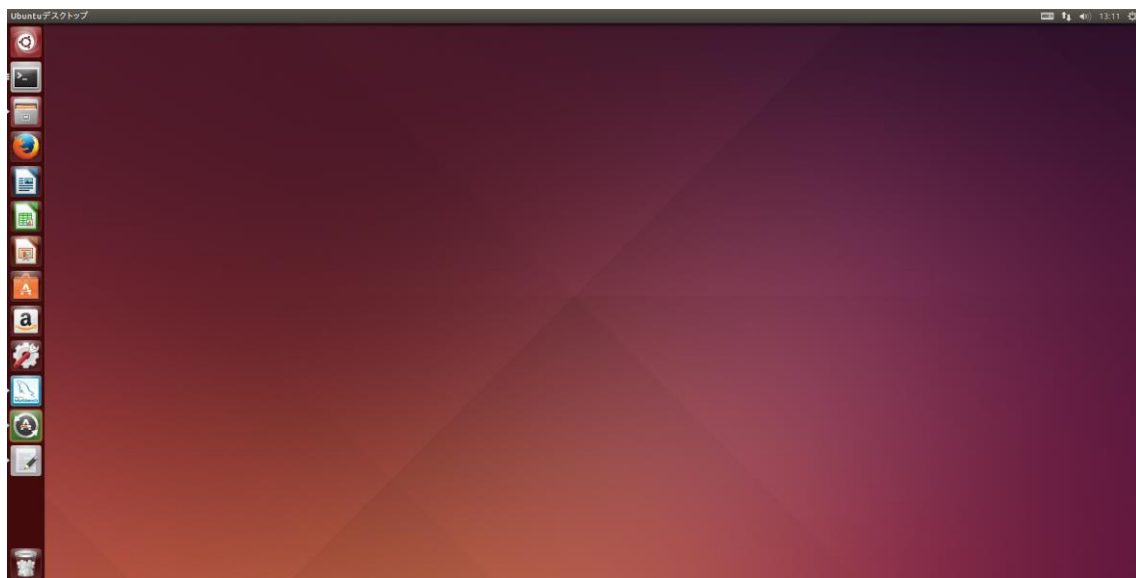


図 5-5 Ubuntu14.04 デスクトップ画面

## 端末の使用方法

本研究では Ubuntu 上での操作には主にアプリケーションのアクセサリ内にある「端末」を使用した。この端末上ではコマンドを実行することによってソフトウェアのインストールやプログラムの実行といった Ubuntu の操作をすべて行うことができる。

端末を開いたウィンドウ内には以下のような表示がされるので、\$ に続けてコマンドを入力し[Enter]キーを押し実行させる。

```
ユーザー名@ホスト名:カレント(作業)ディレクトリ$
```

Linux で使用する主なコマンドは以下の表にまとめた。

コマンド	機能
ls	ファイルの一覧表示
cp	ファイルのコピー
mv	ファイルの移動, ファイル名の変更
rm	ファイルの削除
pwd	カレントディレクトリの表示
cd	カレントディレクトリの変更
cat	ファイルの内容を表示
python	python プログラムの起動
echo	指定した文字列の表示
sh	bash プログラムの起動
mysql	ユーザー名とパスワードを同時に記載することで mysql が起動



### 5.2.2 研究の手順

以下は研究の手順を図にしたものである．手順の説明は以下の図 5-6 研究の手順に沿って行う．なおこの研究の手順に使用したコードの詳細は 5.2.3 実際のコード 内に記載しておく．

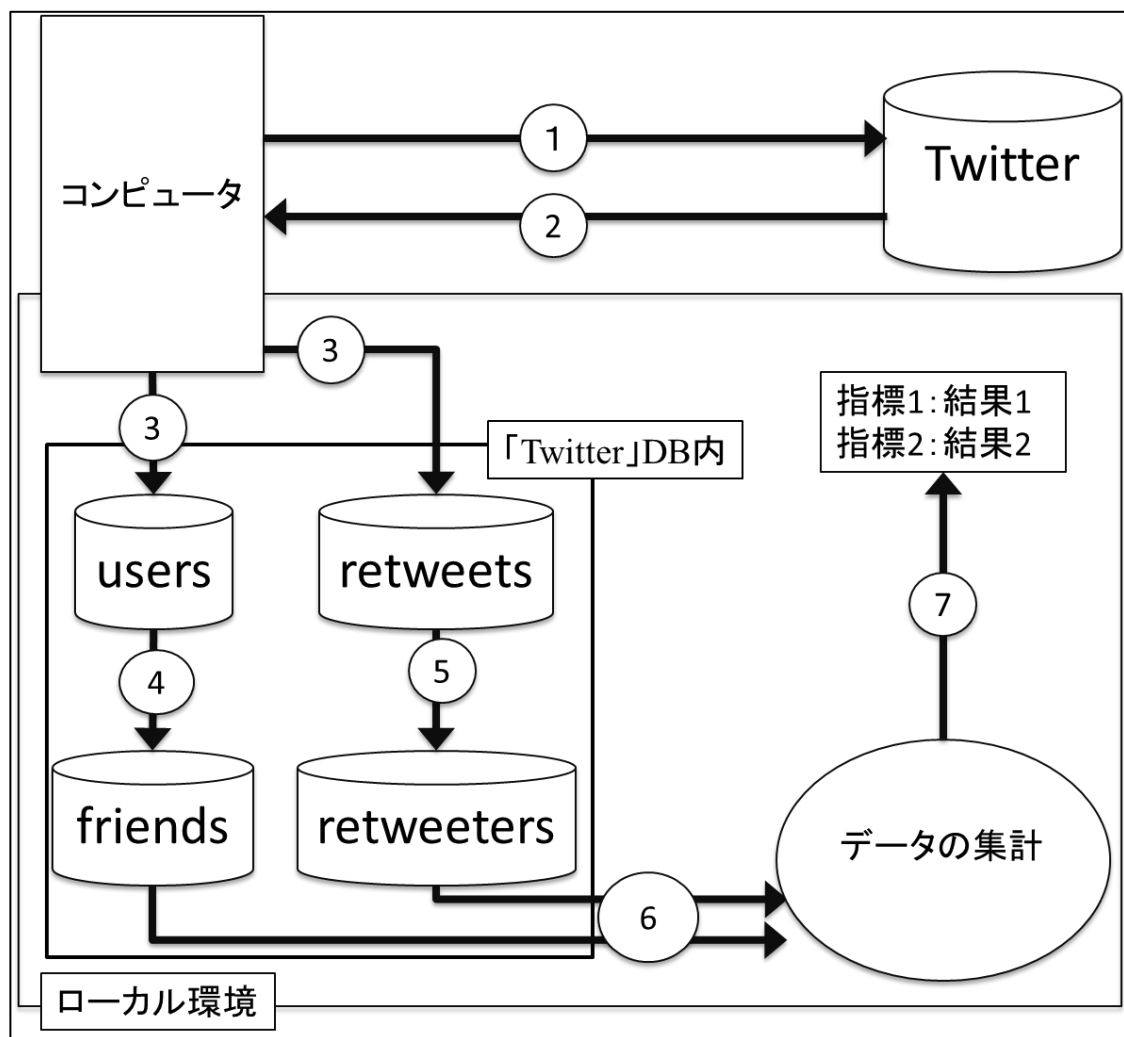


図 5-6 研究の手順

① まず自分のタイムラインに流れてきたツイートの観測を行う．そのためTwitterAPIに従ってTwitterへのアクセスを行った．図5-3 プログラム概要図にあるようにoauth.pyに書き込んであるアクセスキーを使用して端末からsream.pyを使用し，streaminAPI経由でTwitterへアクセスを行った．本研究ではTwitterへのアクセスはTweepyを使用する．

- ② Twitterへのアクセスが認証されると、自分のタイムライン上に投稿されたツイートの成功がJSON形式で端末上に表示される。
- ③ 端末上に表示されたツイートからretweets.pyを使用し、JSON形式のツイート情報の中から必要な情報のみを取り出す。その情報をMySQL上のデータベース(「Twitter」)に作成しておいた「users」と「retweets」のテーブルへ書き込む。
- ④ 「users」テーブル内に保存されているユーザーのIDを1件ずつ取り出しTwitterAPIのREST API経由でそのユーザーのフォローしているユーザーをfriends.pyで取得し、「friends」テーブルを作成していく。「friends」テーブル内にはユーザーのフォローしているユーザーのリストが保存される。
- ⑤ 「retweets」テーブル内に保存されているツイートのIDを1件ずつ取り出しTwitterAPIのREST API経由でそのツイートをリツイートした人のIDをretweeters.pyで取得し、「retweeters」テーブルを作成していく。「retweeters」テーブル内にはツイートをリツイートしているユーザーのリストが保存される。しかし「GET statuses/retweets/:id」のAPIを使用すると、API制限がかかっているため最新の100件のユーザーリストしか取得することができない。
- ⑥ MySQLの中で「Twitter」データベースを選択し、SQL文で必要なデータのみを取り出す。今回必要なデータは指標1と指標2の分母と分子にあたるデータである。抽出に使用したSQL文は以下ようになる。

指標1の分母：リツイートした人で、フォローしている人は調査済みリツイート数

```
mysql > select count(*) from retweets
        join users on users.id=retweets.retweet
        where
        friendsChecked=true
        ;
```

指標 1 の分子：リツイートした人で、フォローしている人は調査済み、かつ、リツイートされた人をフォローしていないリツイート数

```
mysql > select count(*) from retweets
        join users on users.id=retweets.retweet
        where
        friendsChecked=true and
        not exists (select * from friends where friends.user=retweets.retweet
        and friends.friend=retweets.retweeted)
        ;
```

指標 2 の分母：リツイートした人がフォローしている人は調査済み、かつ、誰にリツイートされたかも調査済みのリツイート数

```
mysql > select count(*) from retweets
        join users on users.id=retweets.retweet
        where
        friendsChecked=true
        and retweetersChecked=true
        ;
```

指標 2 の分子：リツイートした人がフォローしている人は調査済み、かつ、誰にリツイートされたかも調査済みのリツイートで、リツイートした人はリツイートされた人とリツイートした人をフォローしていないリツイート数

```
mysql> select count(*) from retweets
      join users on users.id=retweets.retweet
      where
      friendsChecked=true and
      retweetersChecked=true and
      not exists (select * from friends where friends.user=retweets.retweet and
      friends.friend=retweets.retweeted) and
      not exists (
      select * from friends,retweeters where
      retweeters.tweetId=retweets.id and
      friends.user=retweets.retweet and
      friends.friend=retweeters.retweet)
      ;
```

⑦ 結果を求める．集計したデータから指標 1 と指標 2 の割合を導き出す．

今回の研究では Bash を使用し，①，②，③の作業は `retweets.sh` のプログラムを使用し，④の作業は `friends.sh` を，⑤の作業は `retweeters.sh` を使用した．

### 5.2.3 実際のコード

本プログラムで実際に使用したコードを以下に記載する.

stream.py のコード

```
# -*- coding: utf-8 -*-

from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream
from auth import auth

class StdOutListener(StreamListener):
    def on_data(self, data):
        if data.startswith("{"):
            print data
        return True

    def on_error(self, status):
        print status

if __name__ == '__main__':
    stream = Stream(auth, StdOutListener())
    stream.userstream()#タイムラインを取得する場合
```

stream.py のコードの解説

最下部の `userstream()` のカッコ内にダブルクォーテーションで囲んでユーザー名を入力することで取得することができる。

それ以外にこれらのコードを使用することでキーワードによる取得，ランダムサンプリングを取得することができる。

キーワード検索に掛かったもののみ表示する。

```
# stream.filter(track = [キーワード])#検索する場合
```

ランダムサンプリングのみ取得し，表示する。

```
# stream.sample()#ツイートのランダムサンプリングを取得する場合
```

retweets.py のコード

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
import sys, json
from pprint import pprint

def extractUserProfile(user):
    id = user['id_str']
    screenName = user['screen_name']
    statuses = user['statuses_count']
    friends = user['friends_count']
    followers = user['followers_count']
    profileImage = user['profile_image_url'].replace('_normal.', '.')#プロフィール画像（オリジナルサイズ）
    sys.stdout.write("insert into users (id,screenName,statuses,friends,followers,profileImageUrl)
values (%s,'%s',%d,%d,%d,'%s');\n" % (id, screenName, statuses, friends, followers,
profileImage))
    return id
```

続きは次のページへ記す

retweets.py のコードの続き

```
for line in sys.stdin:
    try:
        tweet = json.loads(line)
        #pprint(tweet)
        if 'retweeted_status' in tweet:
            #pprint(tweet)#内容確認（これを有効にして「|less」で実行する）

            retweet = extractUserProfile(tweet['user'])#リツイートした人

            retweetedStatus = tweet['retweeted_status']
            tweet = retweetedStatus['id_str']#ツイート ID
            rcount = retweetedStatus['retweet_count']#ツイート回数
            fcount = retweetedStatus['favorite_count']#ファボ数
            retweeted = extractUserProfile(retweetedStatus['user'])#リツイートされた人

            sys.stdout.write("insert into retweets (id,retweet,retweeted,rcount,fcount) values
(%s,%s,%s,%d,%d);\n" % (tweet, retweet, retweeted, rcount, fcount))
            sys.stdout.flush()
        except ValueError:
            pass
```



friends.py のコード

```
# -*- coding: utf-8 -*-

import tweepy
import json
import sys
from pprint import pprint
from auth import api

for line in sys.stdin:
    userId = line.rstrip()
    sys.stderr.write("checking friends of %s...\n" % userId)

    # フレンドを調べたことを記録する。
    sys.stdout.write("update users set friendChecked=true where id=%s;\n" % (userId))

    try:
        friends = api.friends_ids(userId)
        for friendId in friends:
            sys.stdout.write("insert into friends (user,friend) values (%s,%s);\n" % (userId, friendId))
            sys.stdout.flush()
    except:
        pass (%s,%s,%s,%d,%d);\n" % (tweet, retweet, retweeted, rcount, fcount))
        sys.stdout.flush()
    except ValueError:
        pass
```

retweeters.py のコード

```
# -*- coding: utf-8 -*-

import tweepy
import json
import sys
from pprint import pprint
from auth import api

for line in sys.stdin:
    tweetId = line.rstrip()
    sys.stderr.write("checking retweeters of %s...\n" % tweetId)

    #リツイートした人を調べたことを記録する。
    sys.stdout.write("update retweets set retweetersChecked=true where id=%s;\n" % (tweetId))

    try:
        retweeters = api.retweets(tweetId, 100)
        for t in retweeters:
            #このオブジェクトの中身がわかりにくい。(PyDev のデバッガで調べた。)
            user = t.user
            userId = user.id_str
            screenName = user.screen_name
            statuses = user.statuses_count
            friends = user.friends_count
            followers = user.followers_count

            profileImage = user.profile_image_url.replace('_normal.', '#')#プロフィール画像 (オリジナルサイズ)
```

続きは次のページに記す.

retweeters.py のコードの続き

```
        sys.stdout.write("insert into users
(id,screenName,statuses,friends,followers,profileImageUrl) values
(%s,'%s',%d,%d,%d,'%s');¥n" % (userId, screenName, statuses, friends, followers,
profileImage))#これは重複エラーになるはず
        sys.stdout.write("insert into retweeters (tweetId,retweet) values (%s,%s);¥n" %
(tweetId, userId))
        sys.stdout.flush()
    except:
        pass
```

retweets.sh

```
#!/usr/bin/env bash

while :
do
    python stream.py | python retweets.py | mysql -utest -ppass --force twitter
    sleep 300
done
```

friends.sh

```
#!/usr/bin/env bash

while :
do
    echo "select retweet from retweets join users on users.id=retweets.retweet where
users.friendsChecked=false limit 1;" | mysql -utest -ppass --skip-column-names twitter | python
friends.py | mysql -utest -ppass --force twitter
    sleep 60
done
```

retweeters.sh

```
#!/usr/bin/env bash

while :
do
    echo "select id from retweets where retweetersChecked=false limit 1;" | mysql -utest -ppass
--skip-column-names twitter | python retweeters.py | mysql -utest -ppass --force twitter
    sleep 60
done
```

### 5.3 参考文献

- [1] MakotoHasegawa (2014) 「Ubuntu Japanese Team Wiki」 ,<https://wiki.ubuntulinux.jp/UbuntuTips/Others/HowToUseTerminal>(参照 2014-11-20)
- [2] Matthew A. Russell (2014)『入門ソーシャルデータ 第2版—ソーシャルウェブのデータマイニング』,長尾高弘訳,株式会社オライリージャパン
- [3] TwitterInc (2014) 「Twitter Developer Documentation」 , <https://dev.twitter.com/overview/documentation> (参照 2014-12-22)

# 第 6 章

## 結果・考察

## 6.1 本章について

本章は研究の結果と結果に対する考察を記載する。今回の研究では同じプログラムを使用し2つのアカウントで取得を行ったので、2つの結果から考察を記載する。

## 6.2 研究結果

本研究で使用したプログラムで抽出したデータから、5章に表記してある指標1、指標2の結果を以下に記載する。

下記にある結果は図5-6 研究の手順 ⑥の方法で指標1と指標2の分母と分子を導き出したデータである。図6-1と図6-2はアカウント1、アカウント2それぞれの結果のベン図である。user Stream で取得しているためアカウントのフォロー数による違いは結果に大きく影響を与えるのでフォロー数も記載しておく。

アカウント1		
フォロー数：288		
指標	分母	分子
1	206	121
2	206	82

アカウント2		
フォロー数：787		
指標	分母	分子
1	2123	1216
2	2123	482

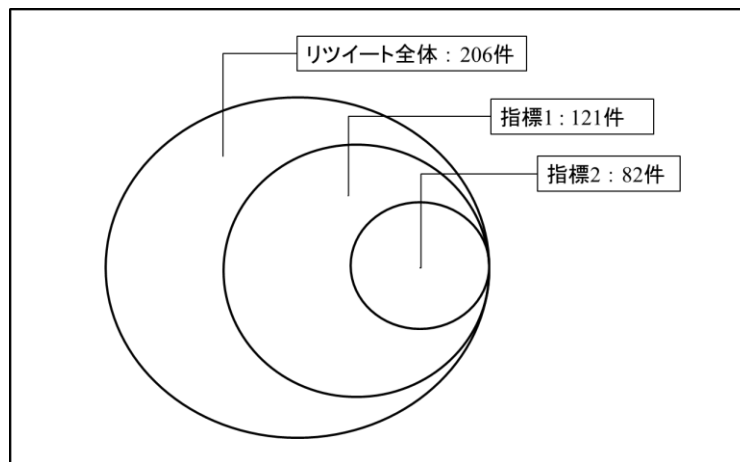


図 6-1 アカウント 1 ベン図

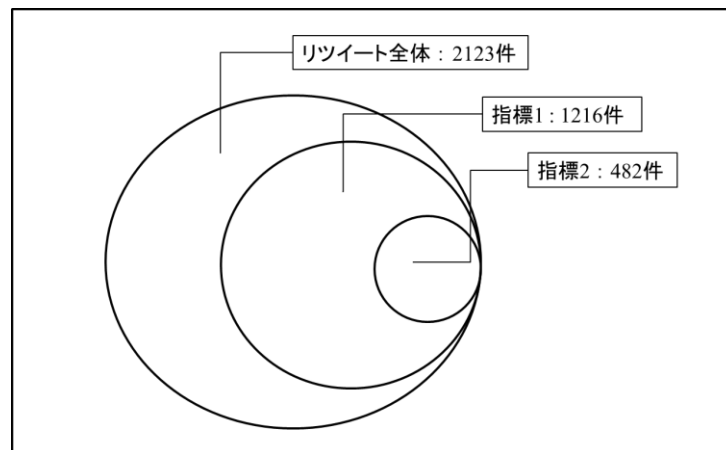


図 6-2 アカウント 2 のベン図

これらの結果をパーセント表記に直すと、

アカウント 1 の結果は指標 1 が 58.7%、指標 2 が 39.8%となった。

アカウント 2 の結果は指標 1 が 57.2%、指標 2 が 22.7%となった。



### 6.3 研究結果考察

6.2 の研究結果から、それに対する考察を記載する。

アカウント 1 から出た研究の結果を確認すると、指標 1 が約 60%、指標 2 が約 40% となった。この結果から確認できることは約 60% ものユーザーがフォローしていないユーザーのツイートも確認し、リツイートを行っていることである。さらに指標 2 の結果をみると、フォローしている人がリツイートをしたものをリツイートしたものではなく、自分で見つけたツイートをリツイートしているユーザーが約 40% もいることが分かった。

リツイートはボタン 1 つで簡単に行える作業であるが、リツイートをする则自分のことをフォローしているユーザーがそのツイートを確認できる状態になるため、ユーザーがそのツイートを簡単に確認するだけではなく発信したいと思う何らかの気持ちや感情が無いとリツイートのボタンは押されないと考える。そのためリツイートされたツイートは、リツイートをしたユーザーがしっかりと確認したうえでリツイートを行っていると考えられる。

つまり指標 1、指標 2 のパーセントが高くなることは、多くのユーザーがフォローしていないユーザーのツイートに対して何らかの感情を持っていることがわかる。つまりフォローしているユーザー以外との Twitter 内に何らかのネットワークが存在している可能性を示す。しかし、逆に指標 1、指標 2 のパーセントが低くなると多くのユーザーは、フォローしているユーザー同士のネットワーク内でツイートを確認していると考えられる。

今研究の結果をみると、フォロー関係でのみ繋がっていると思われていた従来の Twitter ユーザー同士のネットワークの形ではないフォローしていないユーザー同士が何らかの形でつながりあっている新しいネットワークの形がある可能性が十分にあると示された。

### 6.3.1 研究の展望

研究のこれからの展望を記載する。

今回の研究から Twitter のユーザーは自分のフォローしていない人達のツイートを見るネットワークがある可能性が示された。ユーザーがフォローしていないユーザーのツイートを確認する方法としては広告、Twitter 内の「見つける」による発見、リスト等とあるがこのような可能性の中から実際はどのようにユーザー同士がつながりあっているかを突き止めることができれば、この研究の目的であるコミュニティの抽出が可能になり、ユーザービリティの向上に寄与できると考える。

### 6.4 研究方法に対する考察

本研究の方法の考察を記載する。

本研究での2つのアカウントを比べると、アカウント2で調べた結果の指標1が約60%でありアカウント1と非常に近い結果になった。しかし指標2の結果は約20%という結果になりアカウント1とは20%もの誤差がでていることが分かった。これは2人のフォローしている相手の違いやフォローしている人数による差が大きく影響を与えていると思われる、さらに指標2の場合は `retweter.py` で取得しているユーザーが最新100件のみと制限されているため、リツイートされている数が多いツイートを取得している場合は指標2の結果に大きな影響を与えられると思われる。

以下にある図6-3取得したリツイートの人数は図6-4 リツイート回数はアカウント1で取得したリツイート回数とリツイート人数の違いである。なお横軸の数字は tweet ID にあたるので図6-3、図6-4の横軸の長さは同じである。

以下にある図6-5取得したリツイートの人数は図6-6 リツイート回数はアカウント2で取得したリツイート回数とリツイート人数の違いである。なお横軸の数字は tweet ID にあたるので図6-5、図6-6横軸の長さは同じである。

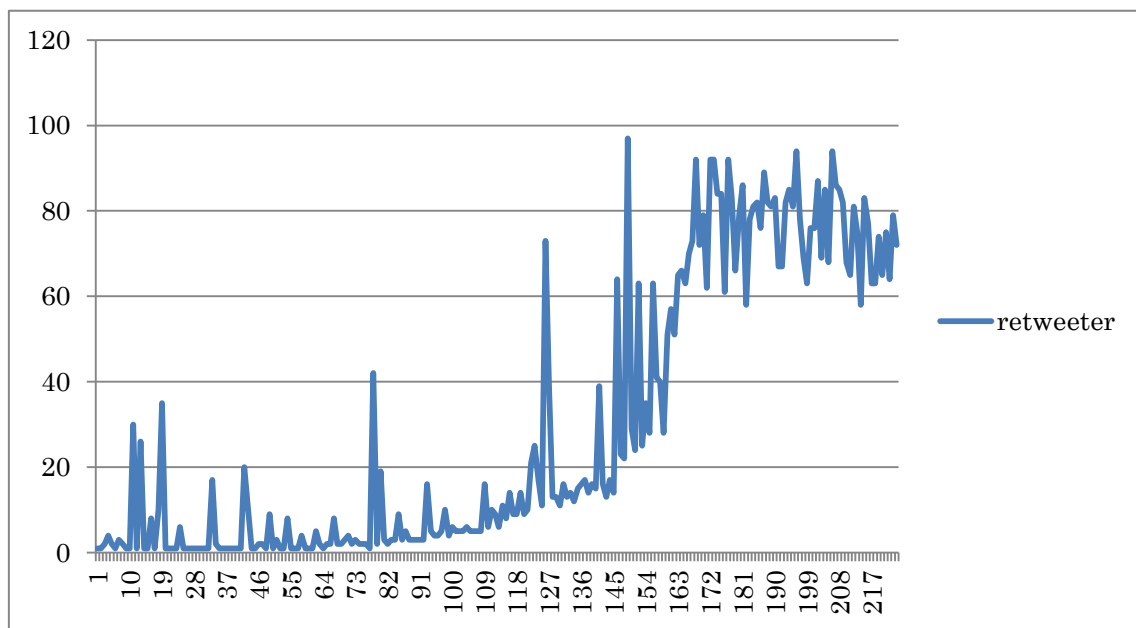


図 6-3 取得したリツイート的人数

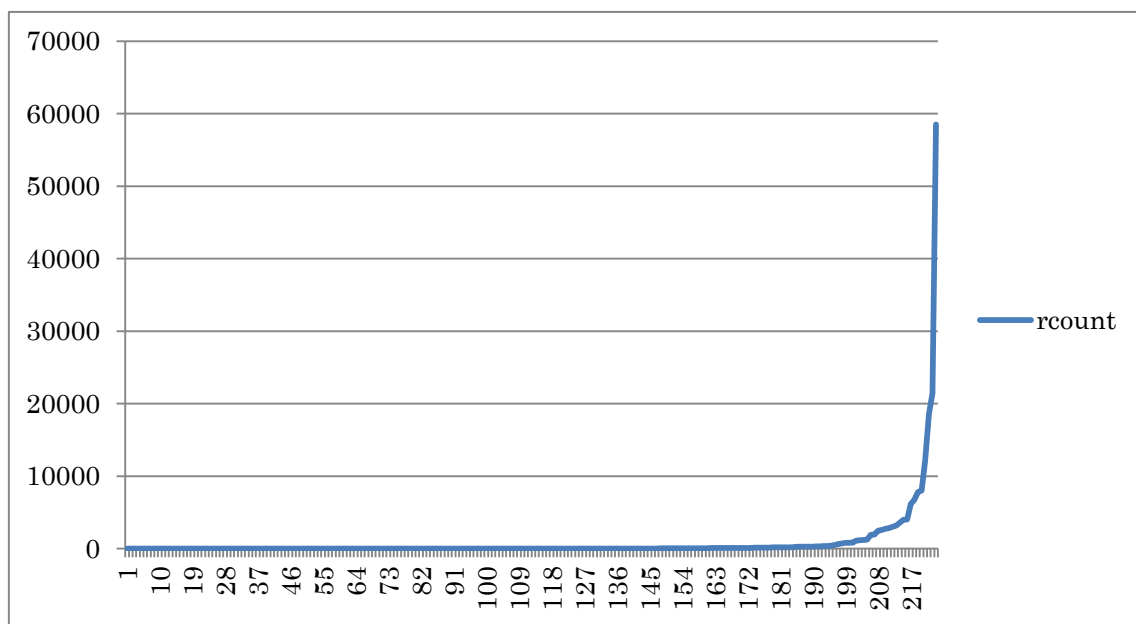


図 6-4 リツイート回数

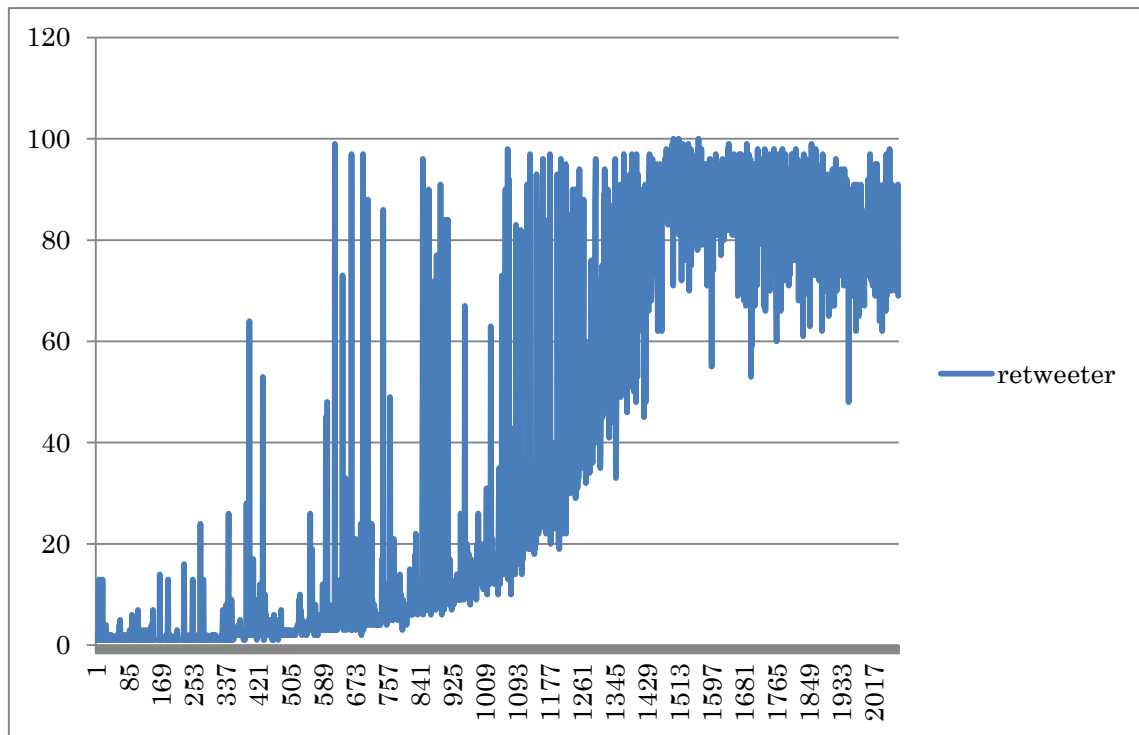


図 6-5 取得したリツイート的人数

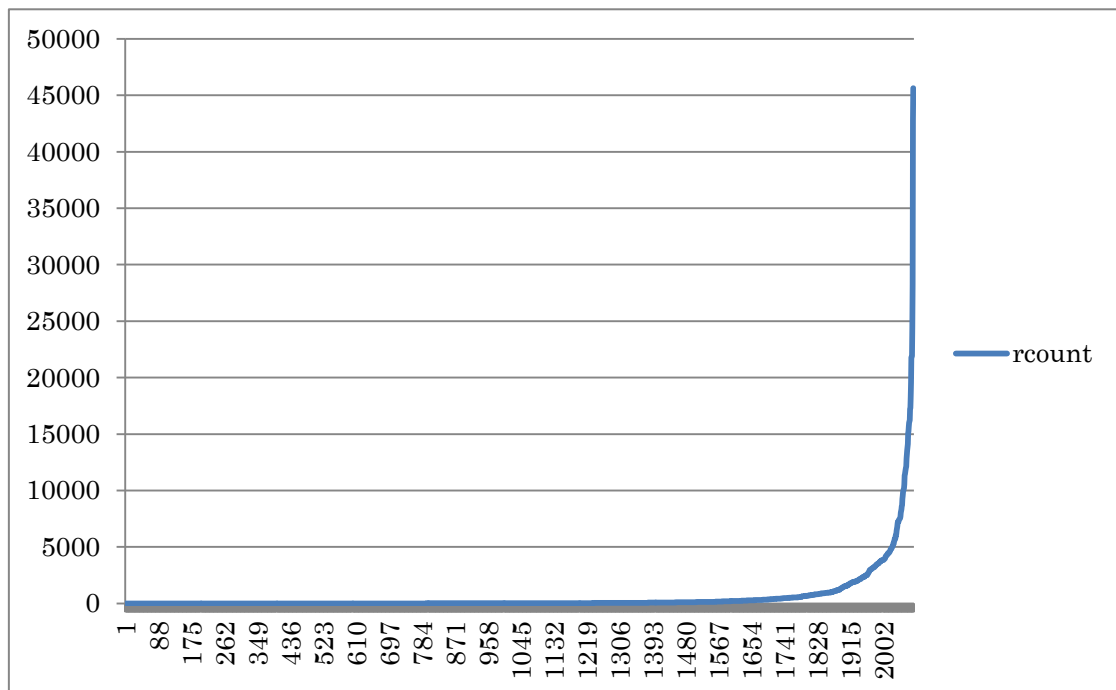


図 6-6 リツイート回数

これらのアカウント1の図 6-3, 図 6-4 とアカウント2の図 6-5, 図 6-6 の違いを確認すると、どちらのアカウントもリツイートが増えているのに取得できている人数は途中から増えていないことがわかる。これが指標2の結果の違いに大きな影響を与えたと思われる。