Vol.2014-DPS-160 No.1
2014/7/24

日常行動のタイムスタンプ生成に特化したライフログ生成システム

佐藤 永欣[1,a)]

概要：センサのデータや、ブログや SNS などの情報をに基づきライフログを作成しようとする動きが始まってひさしい。一方、学生や勤労者のような人は毎日決まった行動をとることが多く、特に労働時間の記録のようなタイムスタンプ自体が重要であるものは、これらのデータをマイニングするよりも毎日の決まった行動の情報に基づく方がよい。そこで、ログイン・ログアウト、部屋のドアの開閉や鍵の開け閉めなどの日常行動に基づくタイムスタンプに特化したライフログシステムを提案する。本システムは裁量労働制を採用しているなどの理由で、比較的ルーズにしか労働時間が管理されていない環境での使用を想定している。プロトタイプとして、特定のマシン群でのログイン・ログアウト操作に基づくタイムスタンプ記録を実装した。本論文では、システムの概要を述べたあと、プロトタイプ実装と評価について述べる。

1. はじめに

さまざまな手段により、ライフログを自動的に生成しようとする研究が始まって 10 年近く経過している。これらの研究は、センサネットワークを利用するもの、モバイル端末と搭載センサを利用するもの、SNS などのテキストデータや位置情報をマイニングするものに大別できる。

本研究では、勤務時間の自動的な記録を主な目的に、タイムスタンプの生成に特化したライフログ生成システムを提案する。本研究では、日常の勤務をいくつかある特定の場所のうちの少なくとも一ヶ所に来て、特定の作業群を行うことと考える。このため、モバイル端末やスマートフォンは利用せずに、特定の場所にある広い意味でのインフラストラクチャ、すなわち PC やエアコン、ドア、照明などの状態を利用することを主に考える。

一般に、勤怠管理の一部として出退勤を記録する場合に、そのタイムスタンプは重要である。労働契約の形態が時間賃金である場合は、賃金計算の基準である。裁量労働制の場合でも、勤務時間がみなし勤務時間と比べて超過している場合、超過分の賃金の支払いが必要な場合がある。

勤怠管理を行う場合、通常は機械式、またはオンラインのタイムレコーダを用いる。しかし、大学のような環境の場合、研究室への到着をもって出勤、研究室からの離脱を持って退勤と考えた方がよい場合がある。事務室に機械式タイムレコーダを設置する、web 上のグループウェアの勤怠管理機能を使用する方法が取られているようである。多くの裁量労働制の場合、出退勤時刻を記録することは強制ではないため、実際には記録が行われていない、記録が行われていても正確とはいいがいたい例がある。

そこで本研究では、勤務に関する日常行動から、出退勤を記録するためのタイムスタンプを十分小さい誤差で自動的に得ることをおもな目的とする。特別な操作をしなくても出退勤を自動的に記録し、法的な係争になった場合を考慮して、改竄や捏造が難しい方法か、改竄・捏造されていても検出できる方法で本システムによる出退勤記録を残すことを最終的な目標とする。

初期の実装として、特定の PC 群へのログインおよびログアウト操作をトリガとした出退勤タイムスタンプの生成をおこなった。すなわち、PC 群のどれか 1 台にログインすれば出勤、PC 群からログアウトすれば退勤とみなす。ただし、長時間かかる計算をさせたまま退勤することや PC を再起動する過程でログアウトすることなどが考えられるので、単純ではない。この他、改竄や捏造に対する部分的な対策を施した実装を行った。

この他、本研究が対象としている岩手県立大学の大部分の部屋では、室内の照明、人感センサの出力、ドアの開閉の状態、エアコンの運転の状態、室温などが集中管理されており、これらの利用も技術的には可能である[*1]。

[1] 岩手県立大学
Iwate Prefectural University, Takizawa, Iwate, 020–0193, Japan
[a)] nobu-s@iwate-pu.ac.jp

[*1] 学術系ネットワークと相互接続されていないネットワークで管理されているため、事務管理上の要求からは不可能であろう

2. 関連研究

ライフログの生成に用いる情報としては、モバイル端末等に搭載されたセンサによるもの [1][2][3]、小型 PC 等により収集したデータを元にブログ記事を生成するもの [5]、センサネットワークによるもの [6]、Twitter[7] のようなマイクロブログや SNS のテキストデータを利用するもの [8] が主である。

モバイル端末と搭載センサを利用するタイプのライフログ生成システムては、GPS や携帯電話基地局を利用した測位結果、加速度センサなどが主に利用されている。マイクロブログや SNS への投稿に基づいてライフログを生成するシステムでは、Twitter のような比較的気軽に投稿できるマイクロブログへの投稿をもとに、位置情報や行動を示す語を検出することによるものがある。そのほかに、Foursquare[9] のような位置情報と関連づけた SNS もライフログ生成の情報源として利用されている [10]。

これらの関連研究では「どのような行動をとったか」「どこにいたか」などに重点が置かれており、タイムスタンプ自体には重点が置かれていない。また、ブログや SNS への投稿を元にする場合やセンサの出力データを元にする場合、正確な行動を常に検出できるわけではないといった問題がある。例えば、加速度センサによる行動推定では、歩いている、走っている、静止しているなどの行動の検出は可能であるが、それらの意味や目的を知ることはできない。また、スマートフォンを中心とした携帯端末を利用する研究では端末の位置情報を利用する例が多いが、バッテリ消費の問題から常に GPS 測位をできないため、位置の更新が遅れがちであったり取りこぼしたりといった問題がある。SNS やブログ等への投稿を元にする場合は、投稿そのものを忘れるとライフログにも記録されない。

これらの点は、勤務時間を正確に記録するためには問題である。すなわち、出勤・退勤の記録が行われない、またこれらの時刻が正確でない可能性があるため、労働者及び使用者双方にとって受け入れ難い。特に、時間制の労働契約の場合は大きな問題である。また、法的な係争が発生した場合を考えると、本システムの記録が法的な証拠として信頼できる必要がある。したがって、各タイムスタンプの時刻が技術的に可能なかぎり正確であること、あとから改竄したり捏造できないこと、改竄や捏造をを防止できない場合では正規のタイムスタンプやイベントかどうかを容易に判断できることが必要である。

これらの点を踏まえて、本研究では、出勤・退勤といったイベントを自動的に検出し、なおかつ取りこぼさない、時刻を可能な範囲で正確に記録する、事後のタイムスタンプの改竄や捏造を困難にし、最低限、改竄されている可能性に気づくこと、などを目標に設計した。

3. タイムスタンプ生成に特化したライフログ作成システム

3.1 基本方針

まず、タイムスタンプ生成に特化したライフログ生成システムを設計するにあたって、要求条件とそれに対する解の方向性について述べる。

(1) イベント発生を取りこぼさない
対象としているイベントは出勤・退勤のようなイベントである。これらの取りこぼしが発生すると、出勤しているのに休んでいることになる、退勤したのに勤務を続けていることになるなどの問題が発生してしまう。

(2) 可能なかぎり正確なタイムスタンプを生成する
勤務時間の記録を主目的としているため、可能なかぎり正確なタイムスタンプを必要とする。本システムが想定している大学などでの利用では時間に縛られない労働契約の形態が多いことから分単位かもっと粗くてもよいが、労働契約によっては高い精度が必要である。

(3) 極力自動化する
出退勤時に行うことは、習慣化されていることが多いと思われる。したがって、習慣化した行動をもとに出退勤のイベントを検出し、自動的にタイムスタンプを生成する。勤務時間の記録が強制でない場合は自動化が有効である。

(4) 事後のタイムスタンプの捏造・改竄防止
勤務時間の記録がおもな目的であるため、事後の改竄は許されない。なんらかの改竄や捏造が可能でありこれらを検出できないとすると、本システムの記録は信用できないということになる。

1 については、建物の一部や備品として存在している人感センサ電気錠の状態、PC のログイン記録などを用いることにより解決する。スマートフォンの搭載センサ類は行動推定の結果の確実性という面ではまだ問題がある。

(2) については本システムを動作させるマシンの時計を NTP などにより正確に合わせておく。本システムは出退勤イベントの検出後、出退勤の状態推定にかかわらず、ただちにタイムスタンプを得ることにより解決する。

(3) については習慣化された操作に、本システムのイベント検知のトリガを仕掛けることにより解決する。これは (1) の問題も概ね解決できる。

(1) から (3) までを総合すると、「可能なかぎり自動的に、いくつかの状況および条件を満たしたらイベント発生とみなし、タイムスタンプを生成する」ということになる。

4 に関しては本システムのみでの解決は難しいと考えられる。考えられる改竄と捏造のパターンとして、次の 4 パターンが考えられる。イベントそのものは残しタイムスタンプ変える、タイムスタンプを残してイベントを現実と違うものに変える、イベントそのものを削除し無かったこと

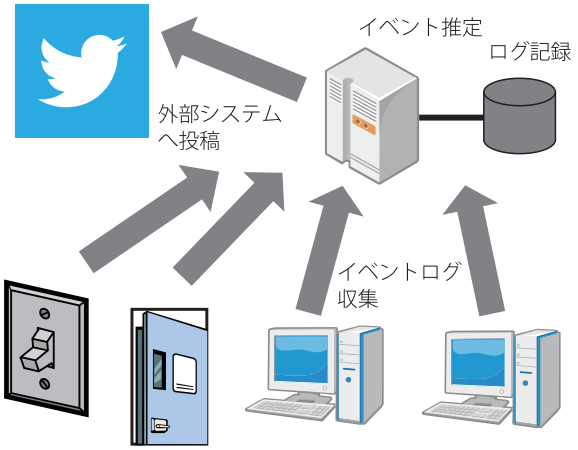

図 1 システム構成と動作の概念

にする、存在していないイベントを作る、である。

本システムが使用する DB などを直接改変することを完全に防止するのは難しい。完全に防止するためにはいくつかの方法が考えられる。ひとつは、DB への全ての書き込みログを保存し、監査し続ける方法である。しかし、これは一番確実な方法であるものの、手間をかけずに実現するのは難しい。この他に、信頼できる第 3 者に、本システムが付けたタイムスタンプと発生したイベントからなるタプルに、第 3 者のタイムスタンプを付加して保証してもらう方法も考えられる。ただし、この方法では、イベントを削除してしまうような改竄や、改竄・捏造したいイベントとほぼ同時刻に行われる改竄・捏造には弱い。

このふたつの方法を比べた場合、後者の方法が実現が簡単と思われる。当面の間、本システムの外部にユーザの出退勤の状態を推定した結果を投稿することで、部分的にこの問題を解決する。具体的には Twitter への投稿を行うことで、タイムスタンプの改竄を発見できるようにする。すなわち、Twitter 投稿時に Twitter 側で投稿に必ず付与するタイムスタンプと、本システムが付与するタイムスタンプの差があまりにも大きい場合、改竄・捏造などが疑われると考える。この方法では、イベントを削除してしまうような改竄は検出できない。また、捏造・改竄したいイベントの発生時刻とほぼ同じ時刻に Twitter への投稿を行った場合や本システムを誤動作させてイベントを発生させた場合なども検出できない。この方法で検出できるのは、後日イベントを改竄・捏造した場合のみである。まずは簡単に実現できることを優先し、本システムで検出が難しいタイプの改竄や捏造については考慮しないことにする。

3.2 設計

次に、提案システムの構成について述べる。**図 1** にシステムの大まかな構成と動作の概念を示す。まず、本システムは大きくわけて次のコンポーネントから構成される。

- イベントログ記録システム。本システムの中心部分であり、イベントログ収集部、イベントログ記録部、状態推定部、状態記録部、SNS 投稿部からなる。
- イベントログ収集用クライアント。端末としての PC にインストールし、ログイン、ログアウトといったイベントを収集、イベント記録システムに送る。

まず、イベントログ記録システムの詳細から述べる。イベントログ記録システムが行動の状態を推定し、推定結果を記録するまでの流れは次のとおりである。

(1) イベントが発生すると、イベントログ収集用クライアントはイベントをイベントログ記録システムに送信する。この際、イベントの種類の他にイベントを発生させたユーザなどの情報もあるのであればまとめて送る。
(2) イベント記録システムのイベントログ収集部は、イベントログ記録部のイベントキューに受け取ったイベントを記録する。この際タイムスタンプも生成して記録する。
(3) 状態推定部はあらかじめ定めたルールに基づき、イベントキューから取り出したイベントを元に行動の状態を推定する。このとき、ルールに定めた条件が満たされない場合、イベントはイベントキューに戻される。
(4) 出退勤の状態が推定できた場合は状態記録部に新たな状態を追記する。
(5) SNS 投稿部は状態記録部に新たな状態が追加されたら、その状態をあらかじめ設定された SNS に投稿する。

イベントが発生した際には、次の例ような XML のメッセージがイベントログ収集用クライアントから送信される。この例は、ログインが発生した場合である。現段階では、ユーザ認証のための情報、及びイベントが発生した PC などのデバイスの情報は、そのまま信用する実装としている。

```
<?xml version="1.0" encoding="UTF-8" ?>
<UserStatus>
  <AuthInfo>
    <UserName>SomeUserIDString</UserName>
    <AuthKey>hogehoge</AuthKey>
  </AuthInfo>
  <Device>SomePcHostName</Device>
  <NewStat>
    <Login />
  </NewStat>
</UserStatus>
```

ここで、UserStatus の各子要素について説明する。

- AuthInfo 要素には認証関連の情報を含める。PC のように誰が使っているのかが明らかなデバイスであれば、その情報を元に本システムでのユーザ名と認証鍵を UserName 子要素と AuthKey 子要素に入れる。エアコンの ON/OFF のように誰が行ったかわからない操作

の場合には、AuthInfo 要素は必要ない。

- Device 要素にはイベントが発生したデバイスの識別情報を含ませる。PC であれば IP アドレス、エアコンの ON/OFF や部屋の鍵、ドアの開閉であれば建物名と部屋番号を使用することを想定している。
- NewStat 要素には発生したイベントを子要素として含ませる。子要素は Login、Logout、DoorOpen、DoorClose、DoorUnlocked、DoorLocked、ACOn、ACOff をいまのところ想定している。

イベントログ収集部は、適当なポートで単にイベントログ収集用クライアントからの TCP 接続を待っている。これは、現在のところ LAN 内でのみ使用することを考えているためである。LAN 外からもイベントログを収集する必要が出た場合、HTTP のようなファイヤーウォールを通過するプロトコルの上に載せるなどの対応が必要である。

イベントログ収集用クライアントはログイン、ログアウトの過程で起動される。起動後ただちにログイン、ログアウトのイベントが発生したことをイベントログ収集部に通知する。通知されたイベントはただちにイベントログ記録部のイベントキューに格納される。イベント状態推定部の状態推定ルーチンは、イベント発生の通知の到着によって起動されるほか、1 分に 1 度定期的に実行される。

イベントキューは、ユーザ、イベントの種類、タイムスタンプ、使用済/未使用のフラグからなるタプルで構成される。ここで、使用済はタプルがキューから排出された状態を示す。排出後もデータが消えないのは、過去に遡った状態の変更が必要になることを考慮しているためである。また、タイムスタンプを生成する判断のエビデンスが要求される可能性も考慮している。

現在のところ、次のようなルールを用いて状態を推定している。

(1) イベントキューから状態の判断に使用されていないイベントを取得する。この際、ログインであれば最古のイベント、ログアウトであれば最新のイベントに着目する。なお、着目されなかったイベントは、連続して同じ種類のイベントが発生している場合は使用済フラグが立つだけで実際には使用されない。例えば、数台の PC に連続してログインする場合である。

(2) 着目したイベントがログイン、ログアウトのいずれかである時は次のように判定する

ログインの時　過去にログインしたことがない、または、過去の最新のログアウトよりも新しい場合、出勤したとみなす。ただし、過去の最新の退勤から、判定対象のログインの時間間隔が短い場合は、過去の最新の退勤を取り消す。すなわち、継続して勤務しているとみなす。

ログアウトの時　ログアウトの時刻と現在時刻を比較し、一定の時間が経過した場合、退勤したものとみなす。一定の時間が経過していない場合は判定を保留する。

(3) 判定が保留以外の結果の場合、対応する処理を行う。すなわち、ログイン、ログアウトの場合はタイムスタンプの記録と SNS への投稿をおこなう。過去の最新の退勤を取り消す判断を行った場合、タイムスタンプの取り消し処理、および SNS への投稿の削除を行う。

ログアウト動作の場合、設定を変更変更してログインしなおす、または再起動するために (再起動のプロセスの一部として) ログアウトすることが考えられる。設定の変更と再起動は頻繁に行われることも考えられるため、一定の時間が経過した後にログインしたような場合は、退勤したと判断したくない。このため、退勤の判定をログアウト直後ではなく、ある程度待ってから行うこととした。

上記の退勤の判断ロジックは必要以上に複雑なようにも思われる。ログアウト直後に退勤と判断し、実際には退勤していないと考えられる状況になった場合に、退勤の判断を取り消す方法も考えられる。しかし、Twitter への投稿を行うため、頻繁な投稿と削除処理を避けるべきだと考えたため、このような複雑なロジックを採用した。今後、Twitter への投稿のみ遅らせるなどの方法も検討する。この場合本システムのタイムスタンプと、それを裏書きする Twitter によるタイムスタンプのずれが大きくなる。

このように、ルールベースでの出退勤の状態推定を行っているが、将来的に、部屋のドアの開閉状態や電気錠の状態を使用する場合は、機械学習ベースの判定方法の採用を検討する必要がある。しかし、正確なタイムスタンプの生成という面からは、ルールベースによる判定が一番確実と考えられる。したがって、ルールベースでの判定に軸足をおきつつ、なんらかの学習をさせ、ルールベースでの判定の結果を補助するのがよいと考えられる。

岩手県立大学の大部分の建物およびほとんど全ての部屋では、次の情報が集中管理されている。

- 各部屋の電気錠の開閉状態。最後に開閉操作を行ったカードキーの識別 ID も管理されている。
- ドアの開閉状態
- 各部屋の人感センサの感知状態
- 各部屋の照明のスイッチの状態。実際の点灯・消灯は人感センサの感知状態と連動する
- 各部屋の温度とエアコンの状態

カードキーの識別 ID は本システムでのユーザ名に変換する必要がある。これらの集中管理されているデータを使用せずにセンサを別途設置する方法も検討するべきであろう。

ログイン・ログアウト以外の PC に関するイベント、例えば、プロセスの起動や終了、CPU 使用率、キーボードやマウス操作のようなコンソールの動きなどを利用するかどうかについても、やはり今後の検討を要する。勤務中に特定の PC 群のうちのどれかを使用していることは確かに

多いが、いずれも使用していないことも考えられるためである。

SNS 投稿部は、現在のところ Twitter への投稿のみを考慮している。本システムが生成したタイムスタンプと発生したイベントの種類をテキストとして投稿する。このとき、Twitter 側が投稿に対してつけるタイムスタンプと本システムが生成したタイムスタンプが若干異なることになる。ログイン、すなわち出勤であれば本システムが生成したタイムスタンプと投稿に Twitter がつけるタイムスタンプはほとんど変わらない。通常は違っていても数秒〜10 秒程度である。一方、退勤の場合、ログアウトから一定時間経過するのを待つため、この時間だけ Twitter への投稿につくタイムスタンプが、本システムが生成したタイムスタンプと比べて新しいことになる。次節で述べるプロトタイプ実装では、3 分間としている。

Twitter の障害を考え、SNS 投稿部は FIFO としている。すなわち、投稿に失敗した場合は FIFO に投稿すべきメッセージが残り、適切な時間が経過したあとにリトライされる。この場合、本システムによるタイムスタンプと Twitter によるタイムスタンプがかなり異なることになってしまう。本システムのタイムスタンプが改竄されていないことの証明ができないことになるため、なんらかの対処を検討すべきである。しかし、Twitter にまったく投稿できない障害がおきることが少ないため、今後の課題としている。

4. 実装

前節で述べた提案システムの一部をプロタイプとして実装した。まず、プロトタイプの実装に使用した環境と言語は次のとおりである。

- PC: Core2Duo E8500 3.56GHz、メモリ 6GB
- OS: FreeBSD 9.1-RELEASE
- DB: SQLite 3.7.17
- 言語: Ruby 1.9.3p448
  - Rubygem SQLite3 1.3.8
  - Rubygem Twitter 4.8.0

今回、プロトタイプとして前節で述べた提案システムを実装した。イベントログ記録部、SNS 投稿部は、それぞれ SQLite のテーブルを持つ。**図 2** にプロトタイプ実装のモジュール構成図をしめす。

前説で述べた設計を検証するためプロトタイプ実装には次のように実装を簡略化している。まず、前節で述べた出退勤イベントを判定するルールはハードコードされている。したがって、全てのユーザで共通である。この点は実用上問題になると考えられるため、今後改善する必要がある。将来的には、判定ルールを記述する言語を導入するのがよいと考えられる。センサ等を使用する際に機械学習の導入も検討すべきであるため、判定ルールを記述する言語を導入する場合、機械学習との整合性をとる必要がある。

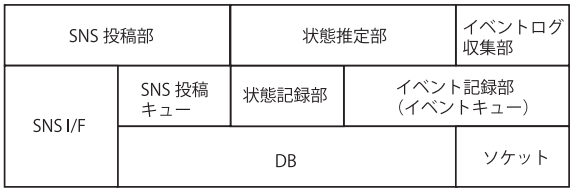

図 2 プロトタイプ実装のモジュール構成図

イベントログ記録用クライアントも Ruby により実装されている。事前に設定された認証情報と発生したログイン・ログアウトのイベントに基づいて、適切なメッセージを適当なポート番号で TCP 接続を待っているイベントログ記録システムのイベントログ収集部に送る。

イベントログ記録システムは 3 個のスレッドに分割されて実行されている。メインスレッドの他には、イベントログ収集部を中心とするスレッド、及び状態判定部を定期的に実行するスレッドである。このため、状態判定部は、イベントログ収集部のスレッドと状態判定部を定期的に実行するスレッドの両方から起動される。状態判定部は二つのスレッドから同時には実行されないように実装されている。

5. 評価

プロトタイプシステムの評価を行った。確実な動作を期してルールベースで出退勤のイベントを判定しているため、判定ミスがないことが確認した上でタイムスタンプの正確性を中心に評価すればよい。

イベントログ収集用クライアントの実行時間であるが、**表 1** に示すとおりであった。この実行時間には、サーバがイベントを受理する処理を行う時間が含まれている。なお time コマンドによる測定のためあまり精度が高い測定結果ではないが、実用には十分短い時間でイベント発生のメッセージを送信できている。

本システムでは Twitter が投稿時につけるタイムスタンプは、投稿そのものを削除しない限り後から変更できないため、これを本システムがつけたタイムスタンプの正しさの保証に使っている。すなわち、本システムのデータベース中のタイムスタンプが Twitter へ投稿のタイムスタンプの時刻と十分近ければ、本システムのタイムスタンプは改竄されていない、とする考えである。ただし、これは、Twitter が正常に動作しており、なおかつ、Twitter のサーバの時刻が正しいことを前提としている。すなわち、本システムのデータベース中のタイムスタンプと、それに対応する Twitter への投稿のタイムスタンプの差が十分小さけれ

表 1 イベントログ収集用クライアント実行時間 (5 回実行)

	実行時間 [s]
平均	0.054
最小	0.050
最大	0.060

表 2 イベントのタイムスタンプと twitter によるタイムスタンプ（出勤，5 回実行）

	タイムスタンプの差 [s]
平均	0.8
最大	1
最小	0

表 3 イベントのタイムスタンプと twitter によるタイムスタンプ（退勤，5 回実行）

	タイムスタンプの差 [s]
平均	222
最大	238
最小	212

ば、本システムのタイムスタンプは信用してよいことになる。また、両者のタイムスタンプが大幅に異なる場合は、Twitter の障害によるものか、改竄や捏造によるものかの区別がつかないが、正常な状態ではないことは確認できる。

このため、本システムが付けたタイムスタンプと、Twitter への投稿に Twitter 側で付けたタイムスタンプの差を確認した。その結果を、**表 2** に示す。イベントログ記録システムを動かしている PC は Strutum 2 の NTP サーバと時刻同期を行っている。また、Twitter のサーバはなんらかの手段で正しい時刻を持っていると仮定している。図 2 に示すように、本システムが付けたタイムスタンプの 0 秒〜1 秒後のタイムスタンプがついている。

時間制の労働契約の場合で、労働時間の記録には 1 分程度の誤差が考えられる。裁量労働制の場合、労働時間の記録には数分〜10 分程度の誤差も許容されている。したがって、基本的には本システムのタイムスタンプには分単位の精度があれば十分と考えられる。時間制の労働契約の場合、8:59 に出勤した場合と 9:00 を若干すぎて出勤した場合では扱いが異なることがある。したがって、基本的には本システムのタイムスタンプの精度は分単位で十分ではあるが、より精度が高いほうがよく、特に出勤時刻が正時をまたぐ場合には秒単位の精度が必要なことも考えられる。

次に、退勤の本システムによるタイムスタンプと Twitter への投稿のタイムスタンプの差を確認した。前節で述べたように、ログアウト動作の場合、設定を変更して再起動などの操作を行うため、退勤の判定をログアウトから一定の時間が経過した後に行っている。現在の実装では、ログアウトから 3 分以上経過していた場合に退勤と判定している。**表 3** に退勤の本システムのタイムスタンプと該当する Twitter への投稿に Twitter がつけたタイムスタンプの差を示す。理論上は 180 秒から 270 秒遅れるはずであるが、測定した限りでは、この範囲に入っている。

6. まとめと今後の課題

本論文では、労働時間の記録をおもな目的とした、可能なかぎり正確なタイムスタンプを生成することを特徴とするライフログシステムを提案した。通常、このようなライフログシステムは、SNS への投稿などをテキストマイニングしたり、センサなどのデータにより行動推定を行うが、本システムは PC や電気錠、照明、エアコンといったインフラストラクチャから情報を収集する。特に、PC へのログインやログアウトといった確度の高い情報を元にルールベースで行動を推定することにより、出勤・退勤といったイベントの取りこぼしを避けている。また、出退勤時刻のようなタイムスタンプは、改竄や消去、捏造の対象となりやすい。これらがされてないことを保証するのは一般的には簡単ではない。本システムは SNS に検出したイベントとそのタイムスタンプを投稿し、SNS に本システムのデータと対応するデータを残し、SNS 投稿のタイムスタンプを傍証として、タイムスタンプの改竄と捏造がされていてもわかることを保証している。

今後の課題は次のとおりである。特定の場所に置いてある PC へのログイン・ログアウトしか考慮していない。さらに PC はネットに接続されている必要がある。出張中やカフェやレストランなどで仕事をすることは考えられるが、どのように扱うべきかは未検討である。また、判定ルールがハードコードされているという問題がある。

参考文献

[1] 片桐郭順, 伊與田光宏, 岡村拓哉,「スマートフォンを利用したライフログの収集」, 情報処理学会第 75 回全国大会講演論文集, Vol.1, pp.289–291 (2013).
[2] 片岡準, 大瀧美香, 渡邉貴之,「位置情報ライフログデータを活用したスケジュール管理支援システム」, 第 75 回全国大会講演論文集, 2013, Vol.1, pp.285–287 (2013).
[3] 大内一成, 土井美和子,「ActivityAnalyzer: 携帯電話搭載センサによるリアルタイム生活行動認識システム」情報処理学会第 30 回ユビキタスコンピューティングシステム研究会, pp.1–8 (2011).
[4] 田中 成典, 中村 健二, 寺口 敏生, 中本 聖也, 加藤 諒,「マイクロブログから抽出したユーザの習慣に基づく行動推定に関する研究」, 情報処理学会論文誌, データベース Vol.6, No.3, pp.83-89 (2013).
[5] 小菅 徹, 吉野 孝,「ライフログを用いたブログ記事自動生成システム BlogWear の開発と評価」, 情報処理学会第 141 回マルチメディア通信と分散処理研究会, pp.1–8 (2009).
[6] 浅川和久, 高橋孝輔, 瀬川典久, 澤本 潤, 山本信次,「センサネットワークを利用した林業活動を支援するウェアラブルシステムの構築 (不均質なライフログからのデータマイニング及び一般)」, 電子情報通信学会技術研究報告, Vol. 109, No.450, ライフインテリジェンスとオフィス情報システム, pp.37–42 (2010).
[7] Twitter, http://twitter.com
[8] 鎌田早織, 坂本寛幸, 井垣 宏, 中村匡秀,「マッシュアップ API を用いた異なるライフログサービスの連携」, 電子情報通信学会技術研究報告, Vol.109, No.450, ライフインテリジェンスとオフィス情報システム, pp.91-96 (2010).
[9] Foursquare, http://foursquare.com
[10] 梶 克彦, 河口信夫 ,「多様な場所への移動をモチベートさせるライフログ活用システム」, 情報処理学会第 75 回全国大会講演論文集, Vol.1, pp.37–39 (2013).