CyberAgent 秋葉原ラボ技術報告

Volume.1

CyberAgent

目次

巻頭言「持続的な研究開発組織に向けて」

秋葉原ラボ 研究室長 福田 一郎

2011 年 4 月にサイバーエージェントの研究開発組織を秋葉原の地に設立して，丸 7 年が経とうとしている．この時に至って初めて秋葉原ラボとしての技術報告書を作成するのは「遅いのではないか？」と思われるところであろう．

言い訳をさせてもらえば，私達のラボは一般的な研究開発組織とは異なり，インターネットメディアを対象とした大規模データの利活用を主軸としており，メンバーも初期は研究者というよりは開発者ばかりであった．そのためアウトプットは全般的にシステムやライブラリという形で，実際にインターネットメディアサービス内で利用されることに重点を置いていた．そのため，まとまった報告書をアウトプットするよりは，それぞれのシステムの実用性などの報告を個々に行ってきた経緯がある．

しかし，会社の規模も拡大し，部署・社員数が増えてくると，秋葉原ラボが何をやっているのかが見えづらくなり，部署内の知見も暗黙的になってきていると感じることが増えてきた．そこで，社内外に知見を共有し，さらには将来の秋葉原ラボへ知見を残していくべく，秋葉原ラボの現在の技術状況を報告書としてまとめることとした．

多少の改変や改称はあれど，7 年間単独の部署として存在しているというのは，移り変わりの激しいインターネット業界にある当社においてはかなり稀な組織であるといえる．しかし，そもそも根幹をなす技術というのはサービスやプロジェクトの盛衰に左右されてはならず，そのためにも研究開発組織を持続的な形で維持することは重要であるという考えのもと，私は秋葉原ラボとその活動に取り組んできた．

この技術報告書はインターネットメディアサービスにおけるデータ活用のために何が必要で，そのために私達が何に取り組んできて，どう発展させてきたのかを著したものになる．報告書という形でまとめることによって新たな示唆や発展を見出し，多くの人々と知見の共有および議論ができるようになり，さらなる秋葉原ラボの持続性につながることを期待している．

秋葉原ラボのあゆみ

秋葉原ラボ 研究室長 福田 一郎

秋葉原ラボは 2011 年 4 月に JR 秋葉原駅前にある秋葉原ダイビル内に開設された Ameba Technology Laboratory を前身組織としている．「なぜ秋葉原に作ったのか？」という問いを受けたことは何度となくある．その問いに対する明確な答えは存在しないのだが，どのようにしてラボができてきたかを見ていくことで，答えに類するものは見えてくるだろう．

Ameba Technology Laboratory ができる 2011 年より以前，2009 年頃にさらなる前身組織であるインキュベーションラボラトリという部署が存在した．そこでは原初的なブログコメントのスパムフィルタシステムやブログ閲覧履歴からの芸能人ブログ推薦システムなどを提供していた．また，同時期にブログ全文検索システムも既に内製で存在していた．

さらに 2010 年 3 月からはモバイルゲームなどを中心にユーザ課金のメディアサービスが増加することを見込んで，ユーザ行動ログを分析する基盤が必要であるという考えから Hadoop/Hive をコアコンポーネントとするログ集計・分析基盤 Patriot の開発が始まった．

これらのデータを活用したシステムを対象に 2010 年夏季学生インターンを実施したところ好評であり，またインターネットサービス企業としてはデータ活用が強みに繋がるという議論から，データ活用を主軸とした研究開発組織があったほうが良いという議論に発展していった．単に組織を作るだけでなく，サービスとは切り離されたところで集中する環境と斬新さを出すためにも場所を渋谷ではない場所にということでオフィスの場所が秋葉原になった．

2011 年 4 月に秋葉原にラボを開設し，2014 年 10 月に Ameba 事業本部の改変に伴い名称を Ameba Technology Laboratory から「秋葉原ラボ」へと変更し，現在に至っている．オフィスも当初は秋葉原ダイビル 8 階にあったが，社員数の増加によって現在は 13 階へと増床移転している．

ラボができるまでの経緯からも，データ集約・集計するデータ分析基盤を中心に機械学習システム構築やデータ分析，当初から引き続きブログの全文検索を中心に各種検索システムを主な対象としている．メディアサービスの発展に伴い，データの種類も増加し，当初のアクセスログや行動ログだけでなく，現在はサービス内の画像や音楽情報データなども対象となってきている．

1. 大規模データ処理システム

現在，インターネット上でサービスを運営する上でデータを活用することの重要性が増している．データの活用によって，ユーザの行動ログや Web サイトへのアクセスログを分析することで適切なサービス改善につなげることや，検索・推薦などのサービスの価値を高める機能を実現できる．また，近年はインターネット上の広告配信などでもデータ活用により品質を向上させることが求められている．

当社では Ameba ブログや AbemaTV など様々なインターネットサービスを運営しており，それらの改善にデータ活用が重要となっている．例えば，Ameba ブログは国内最大規模のブログサービスであり，当社の提供するサービスは多くのユーザに利用されている．このため関連するデータの量も膨大となり，大規模データを効率的に処理するシステムが必要となる．

秋葉原ラボは当社のメディア事業におけるデータの有効活用を目的としており，そのために様々な大規模データ処理システムを開発し，運用している．大規模データ処理の分野は技術の進歩が著しく，またネットサービスは機能改善などが頻繁に行われる．このような変化の多い環境では，システムの開発から運用までを一貫して行い様々な要件に対して柔軟に適応することが重要と考える．

大規模データ処理の分野では様々な技術が提案されており，OSS やクラウドサービスとして実装されているものが多い．そこで秋葉原ラボでは最新の OSS の導入や様々なクラウドサービスとの連携を視野にいれてシステム開発を行っている．また，ストリーム処理など要件が多様で標準的な実装が提供されていないものについては，独自のシステムを研究開発し，実用化している．

本稿では，秋葉原ラボにおけるシステム開発の例として以下を紹介する．まず，当社のデータ解析基盤において最新の OSS やクラウドサービスを活用しながら継続的な改善を実現するための取り組みについて 1.1 章で述べる．1.2 章では，様々なサービスに検索機能を提供するための検索基盤システムについて解説する．最後に，1.3 章では，当社のメディアサービスにおける広告配信などに利用されているストリーム処理エンジンを紹介する．

1.1 データ処理基盤 Patriot における継続的改善の取り組み

梅田 永介，飯島 賢志，斎藤 貴文

概　要　秋葉原ラボでは当社が提供するメディアサービスのデータを有効活用するためのデータ処理基盤の開発・運用を行っている．大規模データ処理の分野は技術の進歩が著しいため，オープンソースソフトウェア (OSS) やクラウドサービスをうまく活用していくことが重要となる．本稿では Patriot を継続的に改善していくための，OSS 活用や他システムとのデータ連携などの取り組みについて紹介する．

Keywords　データ処理基盤，データ管理，OSS

1　はじめに

当社のメディア事業では AbemaTV や Ameba ブログなど多種多様なサービスを提供しており，ユーザの行動ログなど様々なデータが大量に発生している．このようなデータの活用はサービスの改善にかかせなくなっており，秋葉原ラボでは大量のデータを管理, 処理するためのデータ処理基盤 Patriot の開発・運用をおこなっている．Patriot はメディア事業の様々なサービスにおいて，各種レポートの作成やレコメンデーション機能の実現に活用されている．

大規模データ処理の分野は技術の進歩が著しく，またオープンソースソフトウェア (OSS) やクラウドサービスとして利用可能になるものも多い．データを有効活用するためには，最新のツールや手法を効率的に取り込むことが重要となる．そこで，秋葉原ラボでは日々進化する OSS の新しいコンポーネントや最新バージョンを効率的に調査・導入するための環境整備や運用改善を進めている．また，クラウドサービスとの連携を効率的に行うためのデータ転送管理ツールや WebUI などの開発を行っている．

本稿では，最新の OSS の導入やパブリッククラウドなど他システムとの連携により，データ処理基盤 Patriot を継続的に改善していくための取り組みについて記述する．2 節では Patriot の概要を示す．3 節では継続的なソフトウェアアップグレードを行うための取り組みを，4 節では様々な他システムとの連携を実現し，データ品質を改善するためのログ転送管理システムについて説明する．また，5 節では機能改善の例として Massive Parallel Processing (MPP) やカラムナストレージの導入について説明し，6 節ではまとめと今後の取り組みについて述べる．

2　データ処理基盤 Patriot の概要

図 1.1.1 にデータ処理基盤 Patriot の概要を示す．Patriot は Apache Hadoop などの OSS を利用して構築された分散システムであり，当社のメディア事業のデータを収集し，各種レポートの生成やレコメンデーションのためのデータ処理などを行っている．

図 1.1.2 に Patriot のソフトウェア構成を示す．サービスから生成されたデータはデータ転送機能

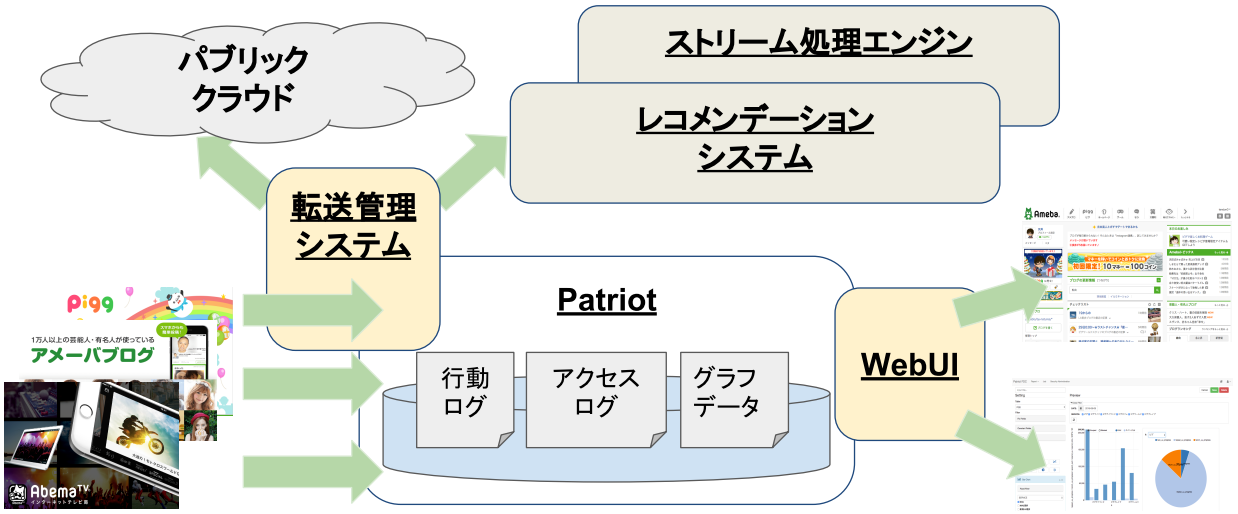


図 1.1.1　データ処理基盤の概要

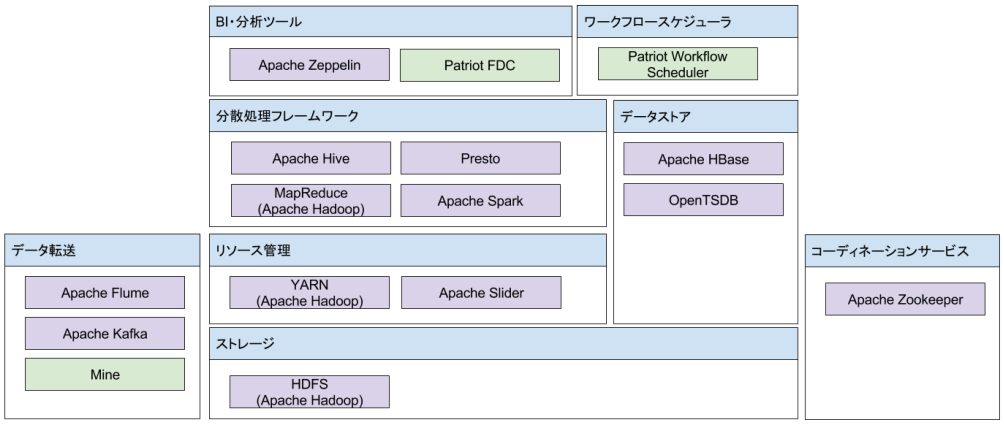


図 1.1.2　Patriot のソフトウェア構成

を利用して収集され，ストレージに格納される．パブリッククラウドやレコメンデーション基盤など他のシステムとの連携を柔軟に行うために，独自の転送管理システム (Mine) を導入している．Mine では Apache Flume のプラグイン機構などを用いてデータの内容に応じたルーティング設定を実現している．

ストレージに格納されたデータはワークフロースケジューラ [2] に従って分散処理フレームワークで処理される．Patriot は社内の様々な部署と連携しているため処理すべきバッチジョブも多様であり，複雑な依存関係を管理する必要があるが，内製のワークフロースケジューラによりこれらの依存関係をワークフローとして管理している．

リソース管理機能は分散処理フレームワークが使用するリソースを管理し，リソースを最大限に活用できるように各処理にリソースを割り当てる．レポーティングのための集計結果はデータストア

に保存され, BI・分析ツールを通じてユーザに利用される．秋葉原ラボでは，Patriot に蓄積したデータを活用するために Patriot FDC という WebUI の開発もおこなっている．

コーディネーションサービスは，ストレージ，データストア，分散処理フレームワークの高可用性を実現するために用いられている．

3 継続的なシステム改善のための取り組み

大規模データ処理の分野は技術の進歩が著しく，継続的に新しいソフトウェアの導入やバージョンアップを行わなければシステムの陳腐化を避けることができない．システムの陳腐化により，先進的な機能への対応が遅れることにもつながる．

Patriot が利用している Apache Hadoop やその周辺システムは，複数のソフトウェアで構成されており，それらの組み合わせによって正しく動作しないこともある．ソフトウェア構成の検証や標準的な動作確認をしたものをベンダが OSS パッケージとして提供しており，これらを利用するのが一般的になっている．Patriot でもベンダ提供の OSS パッケージを利用していたが，一部のソフトウェアのみバグがあった場合のパッチの適用やバージョンアップが難しかったため，OSS パッケージを内製することでこの問題を解決した．また，システムの品質を保証するためのテスト環境を整備することで，新しいソフトウェアの導入やバージョンアップで発生しうる問題を事前に検知できるようにしている．

3.1 OSS パッケージの内製化

ベンダ提供の OSS パッケージを利用する場合，先進機能のバックポートやベンダ独自の修正によりある程度の品質が保証されていたり，OSS パッケージを構成するソフトウェアの組み合わせの検証も行われているという利点がある．そのため，Patriot でもベンダ提供の OSS パッケージを利用していたが, 以下のような問題があり継続的なバージョンアップを行うには不十分な状況にあった．

- ソフトウェアの組み合わせやバージョンが限定されており，新しいソフトウェアを追加で導入したり，特定のソフトウェアのみバージョンアップすることができないため柔軟性に欠ける
- 標準的なテスト環境での動作確認は行われているため，ある程度の品質は保証されてはいるものの，Patriot と同等のデータや動作環境でのテストではないため不具合を十分に検出できていない可能性がある
- ベンダ側の独自の修正によりソースコードが大幅に変更されている場合があり，バグが発生した場合にバグトラッキングシステムや GitHub などで公開されているパッチを適用するのが困難である

これらの問題を回避するため，Patriot では，Apache Bigtop [*1]を利用して OSS パッケージを内製して利用している．Apache Bigtop は OSS ビッグデータコンポーネントのパッケージング，テスト，設定などを行うための OSS である．主要な Linux のパッケージを作成するための設定ファイルや，デーモンプロセスを起動するためのスクリプトが含まれており，Bigtop を用いることで導入が容易なパッケージを作成できる．また，設定ファイルで利用するバージョンをソフトウェアごとに設定でき，パッチの管理もできることから目的に応じたパッケージの作成が可能である．

Patriot の内製 OSS パッケージでカスタマイズされたパッケージとしては，例えば以下のようなものがある．

- Apache Hive 2.1.1
  HIVE-15239[*2]などの処理結果が不正となるバグの修正を適用

[*1] http://bigtop.apache.org/
[*2] https://issues.apache.org/jira/browse/HIVE-15239

- Apache Hadoop 2.8.1
  YARN-5807[*3]などローカルキャッシュを削除しないバグの修正を適用
- Presto 0.190
  Apache Kafka の新しいバージョンとの互換性がない問題[*4]を修正

この他にも，OSS に新機能やバグ対応に関する情報収集・管理を効率的に行うための手法 [5] についても取り組んでいる．

3.2　システムの品質を保証するための取り組み

継続的に新しいソフトウェアの導入やバージョンアップを実現するためには，システムの品質保証が重要な課題となる．そのためには本番環境と同等のデータやシステム構成を用いて動作確認を行うのが理想的である．そこで Patriot では，図 1.1.3 に示すように本番環境と同等のデータをステージング環境にも転送し，本番環境になるべく近い状態でテストができるようにしている．

本番環境と同規模のステージング環境を用意することはコスト的に困難であるが，アクセスログの集計など他のサービスで提供している機能で確認できるものに関してはサンプリングし，アクセス頻度の低い過去のデータの削除などを行うなどの工夫により本番環境と同等のスペックのサーバ構成で，かつ同等の定期バッチの実行を実現している．また，新しいソフトウェア導入やバージョンアップ時の動作確認を効率的に行えるようにしている．

この取り組みにより，ステージング環境での新しいソフトウェアの導入検証やバージョンアップ検証で，ソフトウェアの互換性などの問題を事前に検知することが可能となった．5 節で紹介する Apache ORC を検証する際にもステージング環境で本番環境と同等のデータを用いて動作確認を行い，本番環境に適用する前に問題を検知することができた．

また，各サービスが転送するデータを確認するための環境として小規模なサンドボックス環境を運用している．

4　データ転送管理

当社の Web サービスから生成されるデータはレポーティングやレコメンド機能など，様々な用途で活用できる可能性がある．用途に応じて SLA や管理部門などの違いがあるため，データの転送先を柔軟に設定できる必要がある．

秋葉原ラボでは，この目的のために独自のデータ転送管理システム Mine を開発している．Mine は転送されるデータのヘッダ情報に転送設定を付与することでデータの転送先を制御する．転送設定はデータベース上で管理され，管理ツール（図 1.1.4）を用いてブラウザから変更できる．また転送先ごとに転送データのフィルタリングや外部 API による情報の付与なども設定可能である．

Mine のほぼ全てのデータは Patriot に転送され，一つの Hive テーブルに格納される．他にはレコメンデーションやストリーム処理など準リアルタイムのデータを必要とする集計システムや他部署でデータを必要とするシステム等に転送を行う．

オンプレミスやパブリッククラウドなど様々な環境に対応するために Mine では様々な転送方法をサポートしている．たとえば，オンプレミス環境におけるデータ転送には Flume の利用を推奨している．AWS や GCP などのクラウド環境を利用して Web サービスを構築・運用するケースでは Amazon Kinesis Streams や Google Cloud Pub/Sub などクラウドネイティブなサービスが利用される．スマートフォンの SDK のようなストレージやネットワークに制約が有る環境に対して HTTP のエンドポイントを用意している．

スマートフォン向け SDK などのネットワーク接続が不安定な環境のデータ転送は，転送されるまでの最大遅延時間が決定できないため，データのべき等性を実現することがむずかしい．そこで，

[*3] https://issues.apache.org/jira/browse/YARN-5807
[*4] https://github.com/prestodb/presto/pull/8394

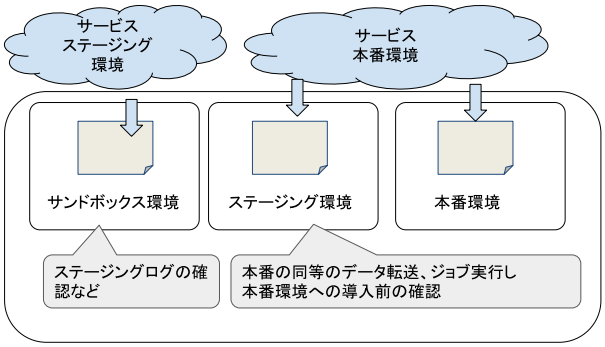


図 1.1.3 Patriot の各環境

図 1.1.4 ログ転送設定の管理

Mine では Google の DataFlow Model [1] などの導入も進めている．

また，Mine ではログの品質を保証するためのログバリデーション機能を導入することでデータの品質改善にも取り組んでいる．

4.1 DataFlow Model の導入

スマートフォン向け SDK などのネットワーク接続が不安定な環境のデータ転送においては，最大遅延時間が決定できないという問題がある．スマートフォンのネイティブアプリにおけるオフライン時に発生するログは一時的にアプリ内でバッファリングされ，オンライン環境になったタイミングでまとめて転送をされるためである．このようなケースでは転送先に届く時刻が，ログが実際に生成された時刻から遅延する．また，遅延の大きさも対象のネイティブアプリがオンラインになるタイミングに依存するため予測が難しい．

この問題を解決するために Google の DataFlow Model [1] の導入を行った．DataFlow Model では，ログが生成された時刻（Event Time）と転送先に到達した時刻（Arrival Time）を分けてあつかう．これにより各ログの遅延時間を把握することができる．また，データを処理する際に Watermark という許容可能な最大遅延時間を設定する．Watermark を基準として遅延したログを集計に含めるかを判別することが可能になり，集計のべき等性を実現することが可能になる．

Mine における Arrival Time は，Mine のストリーミング転送フローに到着した時点で付与される．前述のスマートフォンのネイティブアプリの例で考えると，オンライン環境になり SDK からログが送信され Patriot で管理している HTTP のエンドポイントにログが到達した時点で Arrival Time を付与する．

Arrival Time を考慮した集計の SQL の擬似コードを記す．図 1.1.5 は 2017/1/1 のページビューを集計する HiveQL を簡略的に表したものである．このとき Watermark は 1 時間としている．WHERE 句の前半の部分 (行 4-8) は対象の集計期間に Event Time が含まれるかを判別しており，

```
1  SELECT count(1)
2  FROM test_table
3  WHERE
4     to_unix_timestamp(event_time)
5       BETWEEN
6          to_unix_timestamp("2017-01-01T00:00:00+09:00")
7          AND
8          to_unix_timestamp("2017-01-01T23:59:59+09:00")
9     AND
10    to_unix_timestamp(arrival_time)
11      BETWEEN
12         to_unix_timestamp("2017-01-01T00:00:00+09:00")
13         AND
14         to_unix_timestamp("2017-01-02T00:59:59+09:00")
15    AND
16       action_type = "view"
```

図 1.1.5　Arrival Time を考慮した PV の集計例

後半の部分 (行 10-14) では Arrival Time から導出される遅延が Watermark の範囲内に含まれているかを判別している．図 1.1.5 では 2017/01/01 23:59:59 であるので Arrival Time の範囲は Watermark の 1 時間を加えて 2017/01/02 00:59:59 になる．

4.2　ログバリデーション

データを活用する際にはデータの品質が重要となる．データのフォーマットが統一されていない場合，追加のクレンジング処理などが必要となりデータ活用プロセスが煩雑になる．またフォーマットが統一されていても，予期しないデータが混ざってしまうと同様の問題が発生し，データの活用が困難になる．

Mine では，処理するデータのフォーマットを JSON Schema で管理し, 各ログデータのバリデーションを行うことでデータの品質改善を行っている．JSON Schema は JSON データの内部データを規定するためのスキーマである．JSON Schema を用いることで，ログデータに必須なフィールドや，各フィールドのデータ型やフォーマットを規定できる．

バリデーションを行うことで意図せぬ値が入ったログを集計から除くことができるだけでなく，スキーマ規約に違反したログをエラーレポートとしてサービス側に提供することで，バグを早期に検知することが可能になりデータ改善のサイクルを円滑に進められるような仕組みになっている．

5　探索的データ分析への取り込み

ここでは, Patriot における機能拡張として Massive Parallel Processing (MPP) エンジンとカラム指向型ファイルフォーマットの導入について紹介する．サービスの改善など特定の目的のために集計すべき指標の選定や機械学習のアルゴリズムを考える過程で，発行した処理に対して小さいレイテンシで結果を返すことができれば，対話的に探索的なデータ分析を進めることができる．このように探索的なデータ分析ができれば，より速いサイクルで仮説検証を繰り返すことができ大いに有用である．

Patriot では探索的データ分析を実現する処理エンジンとして，Hadoop クラスタ上で動作する MPP エンジンの 1 つである Presto, および, MPP の性能を引き出す上で大きな役割を果たすカラム指向型ファイルフォーマットである Apache ORC を導入した．

5.1　Presto の導入

Presto は Facebook 社が開発した分散型クエリエンジンであり，中間データをディスクに書き込まずメモリに持つことにより高速な処理ができる．自身でデータソースを持たず, コネクターの機能により複数の外部のデータソースに接続できる特徴がある．Presto クラスタは Presto Coordinator と複数の Presto Worker から構成される． Coordinator はユーザから受け取ったクエリをパース・解析し実行計画を立て，各 Worker が実行する処理をスケジューリングする． Worker はデー

タソースからデータを取得し，それぞれが独立して処理を実行する．最終的に 1 台の Worker が実行結果をマージしてユーザに返す役割を担う．

Presto を利用する際の課題としてリソース管理がある．Presto は CPU コアとメモリともに多くのリソースを消費する．Patriot では，Spark，Hive などの他のジョブも多く動作するため，適切なリソーススケジューリングが必要となる．また，OutOfMemory などの原因で Presto の Worker がダウンする可能性があることも考慮する必要がある．

Patriot では YARN 上で Presto を動作させること（Presto on YARN）でこの問題を解決した．YARN の動的なリソース管理によって，Presto が比較的利用されていない時間帯でのリソース割り当てを小さくすることができ，余剰リソースを他のアプリケーションに有効活用することが可能である．次に，YARN が Presto をコンテナとして起動するため，Presto YARN を起動するサーバのセットアップのみ必要で，実際にコンテナが起動されるサーバに設定を含めた事前のセットアップが不要である．そのため，Presto のアップグレードも容易にすることができる．Presto のプロセスが予期せぬ停止をしたときは YARN が Presto を再起動するため，OutOfMemory などの原因でダウンすることがあっても自動的に復旧することができる．

Presto on YARN は，Coordinator と Worker のうちどれを YARN 上で動作させるかによって様々な構成が可能であるが，Patriot ではステージング環境での調査などを通じて Presto コンテナを同一サーバに複数配置することで起動に失敗する問題などを事前に特定し，安定した構成で本番環境に導入することができた．

5.2 Apache ORC の導入

Apache ORC はパフォーマンスやストレージ効率の向上を図るために開発されたカラム指向型ファイルフォーマットである．ファイル全体を読み込むとディスク I/O がボトルネックとなりパフォーマンスの低下を招きやすいが，必要なカラムのみを読み込むことで大きくディスク I/O を低減できる．加えて Presto はインメモリで処理を行うため，特定のカラムのみをメモリに持つことで使用メモリ量を大幅に低減できる．ORC ファイルはストライプという小さな単位に分割でき，ストライプが各カラムのデータを分割して保持している．また，インデックスデータとして件数，最小値，最大値，合計といった情報があり，これらを利用することでストライプなどの単位で読み飛ばしができ，さらなるパフォーマンスの向上が見込める．

Patriot では Hadoop など各種コンポーネントをバージョンアップし，ORC が利用できるようになったため，ファイルフォーマットを従来の Sequence File から ORC への変換を行った．その結果，Presto，Hive などのジョブに大幅な性能の改善が見られた．

ORC の導入においても第 3 節で説明したステージング環境でのテストを行うことで，HDFS 上に小さなファイルが多数生成され HDFS のメタデータを肥大化させる問題を事前に検知し，ORC を操作する Hive パッケージの修正などを行った上で本番環境に導入できた．

6 まとめ

本稿では，秋葉原ラボにおけるデータ処理基盤 Patriot における継続的改善の取り組みについて紹介した．Patriot では最新の OSS の導入やパブリッククラウドとの連携を効率的に行うために Hadoop パッケージの内製化やログ転送管理システムの開発を行っている．現在は，運用改善をさらに進めるための環境改善や，より多様なデータ活用をサポートするための独自のデータ可視化ツール [4] の開発，Apache Zeppelin の導入を進めている．

当社は様々なインターネットサービスを提供しているため，扱うデータも多様で複雑になりやすい．このため秋葉原ラボでは様々なデータを統合

的に扱うためのツールの開発 [3] や複雑な依存関係を管理するワークフロースケジューラー [2] の改善など分散システムやデータベースに関する研究開発も行っていく予定である.

参考文献

[1] T. Akidau, R. Bradshaw, C. Chambers, S. Chernyak, R. J. Fernández-Moctezuma, R. Lax, S. McVeety, D. Mills, F. Perry, E. Schmidt, and S. Whittle. The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. *Proceedings of the VLDB Endowment*, 8:1792–1803, 2015.

[2] T. Zenmyo, S. Iijima, and I. Fukuda. Managing a complicated workflow based on dataflow-based workflow scheduler. In *2016 IEEE International Conference on Big Data, BigData 2016, Washington DC, USA, December 5-8, 2016*, pp. 1658–1663, 2016.

[3] 善明晃由, 津田均. スキーマ定義に基づく SQL ライクな Key-Value ストアクライアント. 第 8 回データ工学と情報マネジメントに関するフォーラム (DEIM2016).

[4] 善明晃由, 津田均, 田中克季. 可視化のための非構造データの表化手法. 第 9 回データ工学と情報マネジメントに関するフォーラム (DEIM2017).

[5] 大内裕晃，林　晋平，善明晃由，佐伯元司. イシュー上の議論構造の可視化とその理解支援ツール. 電子情報通信学会技術研究報告.

1.2 メディアサービスにおける検索システム基盤の構築と運用

弘瀬 健

概　要　秋葉原ラボでは当社のメディアサービス向けにオープンソースの検索システムである Solr を使用して検索システム基盤を構築し運用している．検索システム基盤の運用方式と検索システム基盤を構成する各コンポーネントについて紹介する．

Keywords　検索システム, 大規模データ

1　はじめに

当社ではさまざまなメディアサービスを提供しているが，サービス毎にコンテンツの内容やユーザのサービス利用パターンが異なっているためそれぞれの検索ニーズが存在する．秋葉原ラボではオープンソースの検索システム Solr[*1]を使用して検索クラスタを構成しサービス毎のニーズに沿った検索システム基盤を構築し運用している．

主に以下の観点により Solr を使用することとなった．

- Solr は内部でオープンソースの検索エンジンライブラリ Lucene[*2]を使用して検索インデックスを作成・管理しているが，設定ファイルにより容易にその検索インデックスを調整することができる．
- Solr の機能である SolrCloud を使用すると容易に検索クラスタを構築・運用することができ，検索・更新負荷が上がった場合でも検索サーバを増設することでスケールすることができる．
- 検索スコアを調整することができるいくつかの機能が用意されており，必要に応じて検索結果の順序を調整することができる．

検索システム基盤では機能毎にコンポーネントを作成してシステムを構築している．サービスと Solr の間に，サービスからのリクエストを受け付ける api コンポーネント，サービス側データソースからデータを取得し加工する batch コンポーネントを置くことでサービス側には検索エンジンを意識させずに検索サービスを提供できている．また，秋葉原ラボ側に Solr 関係の処理を集中させることで Solr を使った検索精度改善や更新効率化といった知見を，検索システム基盤を利用しているサービスに反映しやすくなっている．

以降の節では検索インデックスの管理・更新に関する運用方式，及び検索基盤システムのシステム構成について紹介する．

[*1] http://lucene.apache.org/solr
[*2] https://lucene.apache.org/core/

2　運用方式

検索システム基盤の検索インデックス管理方式と更新について紹介する．

2.1　検索インデックスの管理方式

検索システム基盤では複数のサービスに検索機能を提供している．サービス毎に要件がことなるため，Solr のアップグレードなど影響が大きい運用作業はサービス単位でできることが望ましい．また，特定サービスの検索の負荷が高騰した場合に, 他のサービスへの影響を最小限にすることも必要となる．これらの要件を実現するために, 検索システム基盤では，Solr の SolrCloud 機能を使用して検索インデックスを管理している．SolrCloud ではコレクションという単位で検索インデックスが管理される．コレクションをサービス単位で運用することで，サービス毎に柔軟な検索インデックスの管理が可能となる．

まず，サービス毎にコレクションを管理することで各コレクションで扱うデータを単純にできる．1 コレクションで複数サービスのデータを管理する場合, 保持しているデータが複雑になるためアップグレードの動作確認が難しくなる．特定のデータ型のみ Solr の仕様変更があったような場合，汎用的なスキーマだと対象のデータ型が実際に使用されているかの判断が難しい．

次に，アップグレードの影響範囲を小さくできるため，Solr のアップグレードを行いやすくなる．1 コレクションに複数サービスをまとめてしまうと，アップグレード時にそのコレクションで管理されているサービス全てを一括でアップグレードすることになる．スキーマに関する互換性が保障されないアップグレード時はインデックスの再構築が必要になるが，サービス全てを一括アップグレードする場合，全サービスのデータ再構築が完了するまでアップグレードの内容が反映されない．1 サービス 1 コレクションとしていた場合は，サービス毎にアップグレードを実行でき完了したサービスからアップグレードをサービスに反映できる．アップグレードに問題があった場合の切り戻しもサービス毎にコレクションを分けていた方が容易である．

さらに，SolrCloud ではコレクションの単位でクラスタが管理されるため，サービスごとに利用リソースを分離することが可能となる．これにより，いずれかのサービスの検索負荷が高騰した際の影響を局所化できる．検索負荷高騰に対応するためにはサーバを追加してスケールする必要があるが，利用リソースをサービス毎に分離することで，必要な追加サーバの見積もりを単純化できる．

各サービスのコレクションは，メインとバックアップの SolrCloud クラスタを構築して運用されている．これは障害に対する冗長化と，ローリングアップグレードによるサービス無停止でのアップグレードを可能とするためである．

2.2　検索インデックスの更新

検索インデックスの更新は，後述する batch コンポーネントで実行されるサービス毎の Java プロセスによりそれぞれのデータソースから更新対象データを取得して行っている．更新対象データは，基本的にはブログ記事といったサービス毎のコンテンツ情報であるが，サービスによっては検索スコア調整で使用するための追加情報も検索インデックスに登録している．スコア調整用のデータとしてはコンテンツ毎の読者数や売り上げ額などがあるが, これらはログ解析基盤に送られたサービス毎のログから取得している．

更新対象となるコンテンツ情報はタイムスタンプをもとに判定しており前回のインデックス更新処理実行以降に更新のあったコンテンツ情報が対象となる．コンテンツ情報は数分から数時間以内には検索インデックスに反映される．これに対しスコア調整用のデータはログベースでの連携ということもあり基本的には日次での更新となっている．

スコア調整用のデータ更新処理はコンテンツ情報更新処理とは別プロセスとして実行する．Solr では更新時に commit 処理を行う必要があるが, commit 処理は負荷が高く頻繁に行うものではな

い．上述のように 1 サービスで複数の更新プロセスを並行して実行することがある場合は Solr の auto commit 機能を使い，commit 処理が頻発しないように制御している．

検索インデックスの管理方式の項で述べたように，各サービス毎にメインとバックアップの SolrCloud クラスタを構築して運用しているが，それぞれのクラスタに対する更新処理は別プロセスとして実行している．これは障害に対する冗長化とアップグレード時の運用を考慮してのものである．

3 システム構成

3.1 概要

検索システム基盤は複数のコンポーネントより構成されている. それぞれの概要を以下に示す.

- api: サービスからの検索リクエストを受け付け，Solr 用クエリを作成し node に対してリクエストするコンポーネント.
- node: 検索インデックスを保持しているコンポーネントであり，Solr を SolrCloud モードで起動している.
- batch: サービス毎のデータソースからデータを取得・加工し node に対して更新リクエストを送るコンポーネント.
- zookeeper[*3]: SolrCloud で使用されている分散システムのコーディネートを支援するサービス.batch で使用する更新情報も管理している.

検索システム基盤には検索処理とデータ登録処理が存在する．検索処理では api がサービス側の窓口となりリクエストを受け付ける．受け付けたリクエストをもとに Solr 用クエリを作成しし node に対して検索リクエストを送信する．node では受け取ったリクエストをもとに SolrCloud を使って検索処理を行う．データ登録処理では batch が各サービス毎に用意されているデータソースよりデータを取得しサービス要件に応じて加工する．加工したデータは node に対して送信する．node で受け取ったデータは SolrCloud により Lucene 形式のインデックスファイルとして保管される．

3.2 api

api では主に以下の処理を行う.

- リクエストパラメータの Solr クエリへの変換及び検索リクエスト送信
- 検索クエリログの出力
- node の死活監視

api では各サービスからの検索リクエストを受け付ける. 受け取ったリクエストをもとに Solr 用の検索クエリを生成し，Solr を起動している node へリクエストし検索結果を取得する．取得した検索結果の表示順についてはサービス毎にさまざまなニーズがあり個別に調整を行う必要がある．例えば，コンテンツ作成日の新しい順といった単純なものから，検索を実行したユーザにお勧めのコンテンツを優先して表示など単純なソート機能では実現できないものなどさまざまなニーズが存在する．単純な表示順に関しては Solr のソート機能で対応できるが，特定条件のコンテンツのみ優先表示といったことに関しては Solr の eDisMax[*4]機能を使い検索スコアを調整することで実現する．またお勧めのコンテンツといったユーザに対する推薦の部分に関しては，検索クエリ作成時に api から推薦システム基盤にリクエストして推薦結果を取得し検索クエリに反映することもある．このような検索スコア調整に関する処理は Solr 用の検索クエリ作成時に api で行っている.

また api では各サービスからの検索リクエストを受け取り処理実行した際に検索クエリログを出力している. 出力したクエリログは Flume[*5]を

[*3] https://zookeeper.apache.org/
[*4] https://lucene.apache.org/solr/guide/6_6/the-extended-dismax-query-parser.html
[*5] https://flume.apache.org/

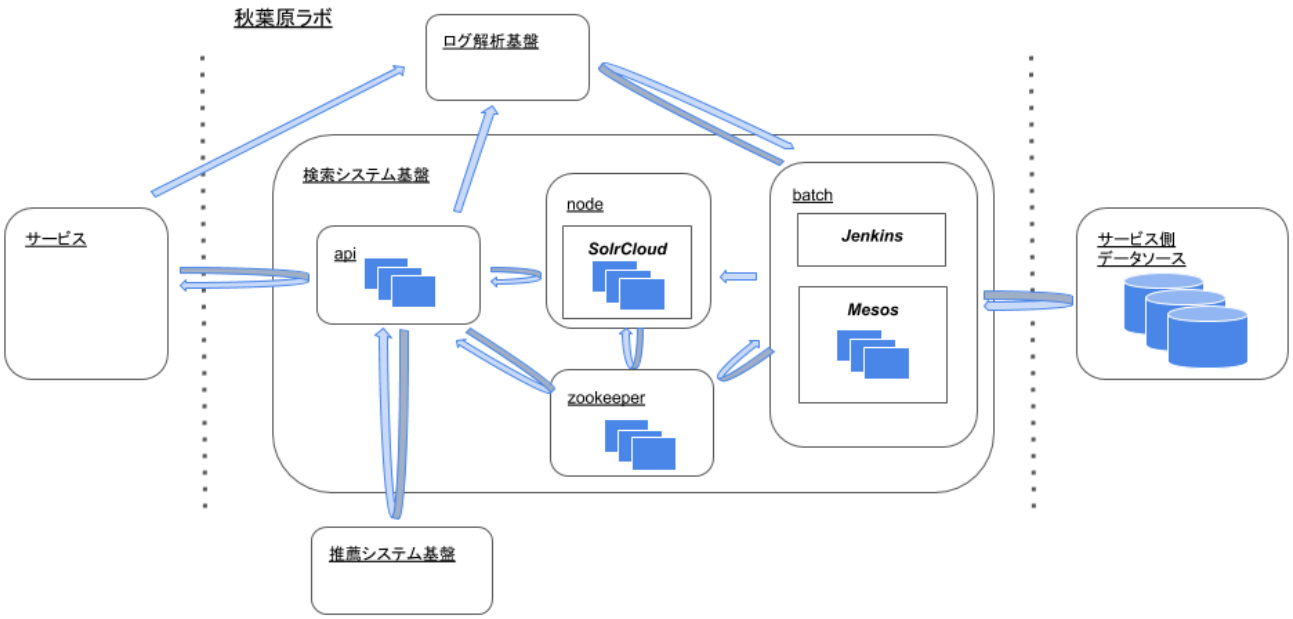


図 1.2.1　検索システム基盤全体図

使ってログ解析基盤に転送し管理している．ログ解析基盤に送られたクエリログは，サービス側の検索画面で利用されるワード補完システムで補完候補キーワードの優先度やキーワードの組み合わせを作成する際に利用される．

検索システム基盤の node では，各サービス毎にメインとバックアップ用 SolrCloud クラスタを作成して運用しており通常はメインの SolrCloud クラスタで検索処理を受け付けるようになっている．ただし，何らかの障害によりメインの SolrCloud クラスタが使えなくなった場合はバックアップの SolrCloud を使用することになる．このメイン・バックアップそれぞれの SolrCloud コレクションの死活監視を api では行っている．死活監視の結果メインの SolrCloud が使えないと分かった場合は，自動的にバックアップの SolrCloud コレクションに対して検索リクエストを送るように api で制御している．

3.3　node

node は検索対象データを管理し検索処理を実行するコンポーネントである．node では Solr を SolrCloud モードで起動しており，検索・更新リクエストは Solr によって処理される．Solr は内部で Lucene を使用しており，検索対象データは Lucene 形式のインデックスファイルとして管理される．node ではサービス毎に複数のサーバを使用して SolrCloud クラスタを構築して運用している．

3.4　batch

batch はサービス毎に用意されたデータソースよりデータを取得し node へ送信する Java プロセスである．取得したデータは node へ送信する前に検索に適した形に加工する．その際にログ解析基盤経由で取得したサービス側ログを使用することもある．batch の Java プロセスは，各システム共通で使用している Mesos[*6] クラスタ上で実行す

[*6] http://mesos.apache.org/
[*7] https://jenkins-ci.org/

る．また Mesos と連携した Jenkins[*7]によって実行スケジュールは管理している．

3.5 zookeeper

分散システムのためのコーディネションサービスであり，Solr が SolrCloud の各種情報を管理するために使用する．api/batch は直接対象となる node の情報を保持していないため，検索/更新時には zookeeper より対象となる node の情報を取得し処理を実行する．

また batch では実行時にタイムスタンプ等を使用して更新対象データを取得するが，batch で使用する更新処理に必要な情報も zookeeper で管理している．

4 まとめ

当社で提供しているメディアサービス向けの検索システム基盤の構成と運用について紹介した．Solr を使うことでサービス毎の検索ニーズを満たしつつスケーラブルな検索システム基盤を構築できた．

1.3 リアルタイムのデータ活用を支援するストリーム処理エンジンの開発

佐藤 栄一

概　要　本稿では，リアルタイムのデータ活用を支援するために開発・運用されているストリーム処理エンジン（開発コード：Zero）について紹介する．本システムは，SQL ライクな独自のクエリ言語「ZeroQL」を事前に記述しておくことで，様々なサービスで発生したログをリアルタイムに処理し，データベースに記録する．記録されたデータは API 経由で数ミリ秒〜数十ミリ秒の低いレイテンシで取得でき，レコメンデーションや広告配信などの配信制御及び精度向上，また，各種指標のリアルタイムのレポーティングなどに活用されている．

Keywords　ストリーム処理，SQL

1　背景

様々なサービスなどから出力されるログをリアルタイムに活用したいというニーズは多い．まずは，速報値のレポーティングのように，リアルタイムの集計値を直接利用するものが挙げられる．また，レコメンデーションや広告配信においては，ユーザーに対するコンテンツの表示回数を制限するフリークエンシーキャップを行ったり，機械学習のモデルにユーザーに関するリアルタイムの素性を含めたりすることで，よりユーザーに最適化したコンテンツを表示することができる．これらの実現には，発生したログ（イベント）をストリーム処理により即時に処理し，データとして利用可能な状態にすることが必要である．

一方で，必要とされるデータに応じてシステムをそれぞれ開発，継続運用するには，多大な労力を要する．汎用的なストリーム処理システムが必要である．

2 節では，システムの設計について述べる．3 節では，システムの実装について述べる．4 節以降，まとめと今後の展望を述べる．

2　システムの設計

はじめに，本システムではイベントの転送について次のような前提を置いている．

配送保証　イベントの転送過程において，欠損を防ぐために，再送が行われることがある．結果的に複数回，イベントが届くことがある．

配送順序　イベントの到着順序は保証されていない．イベントが時刻順など何らかの意味のある順序で届くとは限らない．

この前提から，イベントの重複や到着順序に影響を受けない処理を考える必要がある．詳細は後述するが，前者は重複を除外する仕組みを設けること，後者は可換モノイドというデータ構造を用いることにより解決している．

2.1　ZeroQL

ストリーム処理を汎用的なシステムとして実装する場合，どのような処理を行うのか，簡潔に記

述できる必要がある．また，当社では主に JSON 形式のログが扱われており，本システムにおいても JSON 形式のイベントから必要なフィールドを取り出すなどの変換操作を簡潔に記述できなければならない．

このため，本システムでは SQL に類似した言語「ZeroQL」を独自に定義した．また，JSON ドキュメントの変換を行うコマンドラインツールとして jq[*1]がよく知られており，同言語を ZeroQL に組み込む形で採用した．ただし，本システムは Java 言語で記述しており，C 言語で実装されたコマンドラインツールである jq はそのままでは利用に適さないため，JVM 上で動作するライブラリとして jackson-jq[*2]を実装して用いた．

ZeroQL の記述例をリスト 1.3.1 に示す．以降，本項では ZeroQL の構文について簡単に説明する．

```
SELECT
  count(*) AS total,
  sum(`if .type == "click"
       then 1 else 0 end`) AS click,
FROM flume(`.time`)
  DISTINCT ON `.uuid` FOR 10m
WHERE `headers.service == "foo"`
GROUP BY `{
  user_id: .user_id
}`
PARTITION BY user_id
INTERVAL 1d
```

ソースコード 1.3.1 ZeroQL の記述例: user_id 毎のログの件数とそのうち type が click である件数をそれぞれ total, click として集計している．

SELECT ... FROM ... 文

SELECT 以降には，必要な集約関数を指定する．集約関数の詳細については，2.2 節を参照されたい．

FROM 句には，データの取得元を指定する．続く括弧内には，イベントを処理する際に基準となる時刻を jq の式で指定する．本システムでは，処理を行っている時点での時刻に限らず，イベントの発生時刻など，任意の時刻を基準に処理を行うことができる．この基準時刻は，後述する [GROUP BY] INTERVAL 句や一部の集約関数で使用される．

また，DISTINCT 句を指定することで，重複して届いたイベントの重複排除を行うことができる．再送などにより同一のイベントが複数回届く可能性のある状況で count(*) など冪等[*3]ではない集約関数を用いる場合には必要となる．なお，重複排除はすでに処理したイベントの ID などを記録しておくことで行うが，記憶容量の制約から保存期間を続く FOR 句で指定する．

GROUP BY ... 句

GROUP BY 句には，オブジェクトを生成する jq の式を指定する．生成されたオブジェクトの等価性に基づきグルーピングの処理が行われる．また，複数のオブジェクトを生成する jq の式を指定することで，単一のイベントを複数のグループに計上することも可能である．これはイベントに配列として含まれている情報などでグルーピングしたいときに有用である．

[GROUP BY] INTERVAL ... 句

[GROUP BY] INTERVAL 句は時刻の処理に特化した GROUP BY 句である[*4]．集約関数による処理を一定の期間区切りで行うよう指定でき，例えば，全期間 (NONE)，1 日毎 (1d)，1 時間毎 (1h)，10 分毎 (10m) などを複数指定することができる．指定しない場合は，暗黙的に全期間での集約となる．この処理は FROM 句で指定したイベントの基準時刻をもとに行われる．また，INTERVAL を指定したテーブルでは，結果の取得時に期間指定ができ，指定した期間の結果をまとめて取得したり，指定した期間でさらに集約した結果を取得できる．

[*1] https://stedolan.github.io/jq

[*2] https://github.com/eiiches/jackson-jq

[*3] 対応するモノイド $(S, \bullet)$ が $a \bullet a \neq a$ $(a \in S)$ という意味で使用している．モノイドについては後述する．

[*4] [GROUP BY] INTERVAL の GROUP BY 部分は省略可能である．

2.2 集約関数

本システムでは，以下の集約関数を提供[*5]している．実装上，本システムでの集約関数は実際には集約処理をしておらず，イベントを後述する可換モノイドに変換して出力するだけの関数である．

- `count(*)`, `count(expr)`: 件数を求める．後者は `expr` が `null` ではない件数を求める．
- `sum(expr)`: `expr` で指定した数値の合計を求める．
- `min(expr)`, `max(expr)`: それぞれ，`expr` で指定した値の最小値，最大値を求める．
- `latest(expr)`, `oldest(expr)`: `expr` で指定した値の最新値，もしくは最も古い値を求める．
- `latest_n(expr, n)`: `expr` で指定した値の最新の `n` 件を求める．
- `latest_set_n(expr, n)`: `expr` で指定した値の重複を除いた最新の n 件を求める．
- `reduce(expr, reduce[, postprocess])`: `expr` で指定した値を `reduce` で指定した式で集約した値を返す．これは jq により処理を自由に記述できるようにした集約関数[*6]であり，例えば，`reduce(1, `. + $value`)` は `count(*)` と同じ結果を返す．

なお，一部の集約関数は，`FROM` 句で指定した時刻を基準に動作する．

2.3 モノイド

集合 S，その上の二項演算 $\bullet : S \times S \to S$ が与えられた時，次の条件を満たす組 $(S, \bullet)$ をモノイドという[*7]．ただし，$a, b, c \in S$ とする．

結合則 $(a \bullet b) \bullet c = a \bullet (b \bullet c)$

単位元の存在 $\exists e \in S, e \bullet a = a \bullet e = a$

また，この定義に加え，次の交換則を加えたものを，可換モノイドという．

交換則 $a \bullet b = b \bullet a$

例えば，実数全体の集合を $\mathcal{R}$ とするとき，$(\mathcal{R}, +)$, $(\mathcal{R}, \times)$ などが可換モノイドである．集合 S も演算 $(\bullet)$ も自由に定義できるため，結合則と交換則さえ満たせば，単位元として `null` などを集合 S に加えることで，実用上は容易に可換モノイドを構成することができる．例えば，$\mathcal{R}$ に `null` を加えた集合 $\mathcal{R}'$ を定義したとき，`null` が単位元となるように両辺のうち大きい方を返す二項演算 `max` を定義すれば，$(\mathcal{R}', \texttt{max})$ は可換モノイドである．

イベントを集約関数により可換モノイドに変換して扱うことで，イベントの到着順序に関わらず，また，複数のサーバーで先に別々に集約しても，最終的な結果は変化しない．このような処理の方法は，処理のタイミングなどが異なるだけで，バッチ処理でよく知られている MapReduce に他ならない[*8]．

2.4 データモデル

本システムでのデータの概念図を表 1.3.1 に示す．Record Keys は ZeroQL の `GROUP BY` 句によって決まる．また，Record Values の col1, col2 は，ZeroQL の `SELECT` 以降の集約関数のリストによって決まる．Record Keys は，内部では 2 つの部分 (hash-indexed, dynamic) に分かれており，ZeroQL で `PARTITION BY` 句で指定したものが，hash-indexed 部分になり，残りが dynamic 部分として扱われる．hash-indexed 部分は，結果の取得時など，データにアクセスする際に必須である．

Interval と Timestamp は，`INTERVAL` 句により

[*5] 厳密には，現在の実装では latest_long(), latest_json() など集約関数末尾に型名を表す接尾辞が付く場合がある．将来的には統合され，接尾時が削除される予定である．
[*6] 関数型言語によく見られる reduce や fold 関数に近い．
[*7] Haskell の Monoid 型クラスなど，一部の言語ではモノイドに対応する仕組みが用意されていることがある．
[*8] 集約関数によりイベントからモノイドへ変換する操作が Map，データベースへの書き込み時などに行うモノイドの演算 $(\bullet)$ が Reduce に対応する．

表 1.3.1 データモデル

Record Keys (hash-indexed)		**Interval**	**Timestamp** (sorted)	**Record Keys** (dynamic)	**Record Values**	
gk1	gk2				col1	col2
foo	a	1 day	2017-12-11	{col1: a, col2: b}	27	16
foo	a	1 day	2017-12-11	{col1: a, col2: c}	31	94
foo	a	1 day	2017-12-12	{col1: a, col2: b}	47	61
foo	a	1 hour	2017-12-11T10	{col1: a, col2: b}	20	11
foo	a	1 hour	2017-12-11T11	{col1: a, col2: c}	31	94
foo	a	1 hour	2017-12-11T12	{col1: a, col2: b}	7	5
foo	a	1 hour	2017-12-12T07	{col1: a, col2: b}	47	61
bar	c	1 day	2017-12-11	{col1: a, col2: b}	23	29
bar	b	1 day	2017-12-11	{col1: a, col2: b}	42	47

有限の時間区切りを指定した場合のみ，使用される．

3 システムの実装

本システムは，図 1.3.1 に示すコンポーネントにより構成されている．一部のコンポーネントは，必要とされる要件やクラウド環境に応じて切り替えたり，併用したりしている[*9]．

Event Processor イベントを受け取り，モノイドへの変換後，Monoid Storage へ書き込みを行うサーバーである．

Query Server Monoid Storage からモノイドを取得し結果を返す．Event Processor と同一のソースコードで動作しているため必ずしも分ける必要はないが，負荷の傾向が異なるため，運用上の都合上サーバーを分けている．

Monoid Storage モノイドの記録を行う．Apache HBase などを使用しているが，詳細は 3.2 項で述べる．

Dedup Server イベントの重複排除のために，処理済みのイベントの記録を行う．Redis を使用する．

Cache Server Monoid Storage の負荷軽減及びレイテンシの削減のため，モノイドのキャッシュを行う．Redis を使用する．

Config Registry ZeroQL やキャッシュ設定など，テーブル設定の保存を行う．Apache ZooKeeper や MySQL を使用する．

Discovery Service Cache Server, Dedup Server の発見を行う．Apache ZooKeeper や Kubernetes の Endpoints API を使用する．

3.1 イベントの処理

Event Processor の処理の流れを図 1.3.2 に示した．まずは，Apache Flume [*10] など[*11]のストリーム転送ミドルウェアからイベントを受け取る．その後，ZeroQL で指定した集約関数によってモノイドに変換される．このとき，すでに処理済みのイベントかを確認し処理済みの場合は，イベントは破棄する．モノイドは書き込みバッファに追加され，最終的にストレージに書き込みが行われる．書き込み後，対応するキャッシュエントリが存在する場合は，無効化される．

[*9] 依存するライブラリの競合が避けられないこともあり，クラスローダーを分離したプラグイン機構を実装している．
[*10] `https://flume.apache.org/`
[*11] 他に，Apache Kafka (`https://kafka.apache.org/`)，Amazon Kinesis (`https://aws.amazon.com/kinesis/`)，Google Cloud Pub/Sub (`https://cloud.google.com/pubsub/`) に対応している．

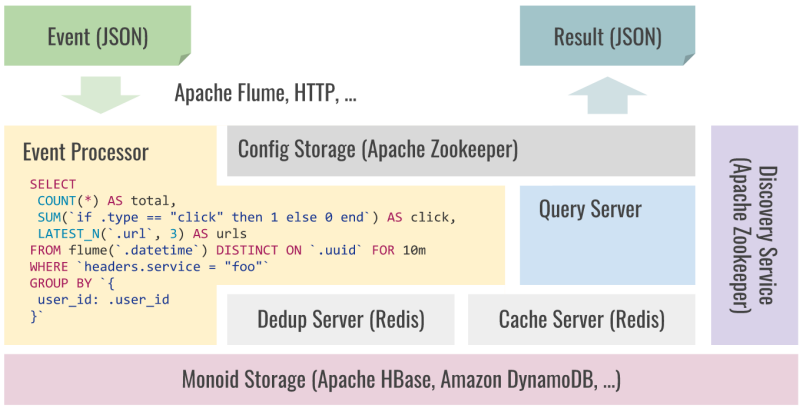


図 1.3.1　コンポーネント図

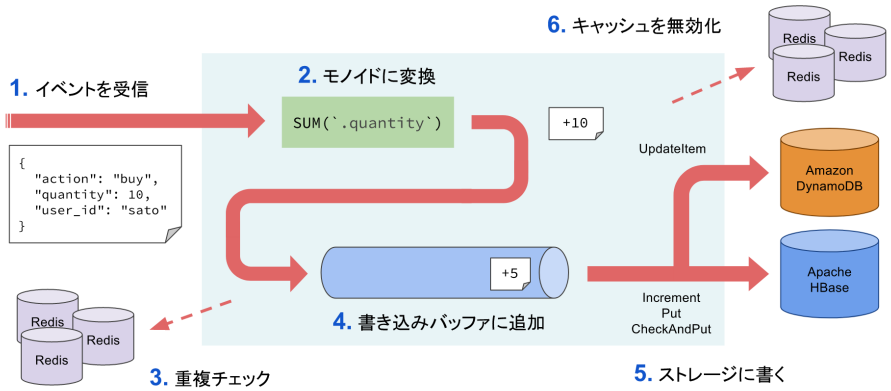


図 1.3.2　イベントの処理の流れ

3.2　Monoid Storage

Monoid Storage は，集約関数により変換されたモノイドを保存するデータベースである．このデータベースは大量のランダムアクセスを低いレイテンシで処理できる必要があり，現在は Apache HBase, Google Cloud Bigtable 及び，Amazon DynamoDB を使用している．

Apache HBase / Google Cloud Bigtable

Apache HBase[*12]は，Bigtable[*13]をモデルに実装されたスケーラブルな分散データストアである．また，Google Cloud Bigtable[*14]は，Google により提供されている NoSQL データベースサービスであり，HBase と互換性のあるインターフェースで使用することができる[*15]．

表 1.3.2 に本システムのデータモデル（表 1.3.1）と HBase のスキーマの対応関係を示した．HBase のセルには，値として各モノイドを `byte` 列にエンコードしたものを格納している[*16]．また，`INTERVAL` に応じて異なる TTL を設定したいこともあり，この場合は `INTERVAL` 毎にカラムファミリーを分割している．

行キーの先頭には，Record Keys (hash-indexed) のうち一部のキーのハッシュ値を 1 `byte`

[*12] `https://hbase.apache.org/`
[*13] `https://research.google.com/archive/bigtable.html`
[*14] `https://cloud.google.com/bigtable/`
[*15] Cloud Bigtable の API を直接使用することも出来る．実装上，HBase 1.x の非同期ではない Java API を避けたかったため，本システムでは Cloud Bigtable の API を直接使用している．
[*16] latest() 関数により出力されるモノイドなど，一部のモノイドはセルの値だけでなくタイムスタンプも使用して保存している．

表 1.3.2 データモデルとスキーマの対応関係

データモデル	HBase / Cloud Bigtable	DynamoDB
Record Keys (hash-indexed)	行キー	パーティションキー（または，部分的にソートキー）
Interval	行キー（または，列ファミリー）	ソートキー
Timestamp	行キー	ソートキー
Record Keys (dynamic)	列 (Qualifier)	ソートキー
Record Values の列	列 (Qualifier)	属性 (Attribute)
モノイド値	値，タイムスタンプ	属性の値 (Value)

付与している．これは，データを均一に分散させるためであるが，一部のみを使うのは，アクセスパターンを考慮し，例えば，ユーザー毎にデータが連続している方がアクセスの局所性が高くなるためである．

モノイドの書き込みには，`Increment`，`Put` を使用するが，一部のモノイドの書き込みでは，セルに対する `CheckAndPut` 操作が必要になる．この場合，まずはセルのデータの `Get` を行う．`Get` により取得したモノイドと，これから書き込む内容とモノイドの演算を行った結果のモノイドを `CheckAndPut` により書き込むが，このとき HBase のサーバーでは，既存の値と比較し，その間に別の変更が行われていない場合に限り，`CheckAndPut` を処理する．変更が行われていた場合は，`Get` からやり直す．複数のサーバーから同時に同じセルに対して `CheckAndPut` を行った場合，成功するのはどれか一つであり非効率であるが，本節の後半で紹介する書き込みのバッファリングを行うことにより，同じ行への書き込みが発生しづらくなっている．例えば，ユーザー毎の集計等を行う場合には，カーディナリティが高く，書き込む行がよく分散するため競合はそもそも稀であり，一方で，カーディナリティの低い `GROUP BY` を行う場合には，書き込みのバッファリングが良く効くため，同じ行への書き込み回数が少なくなり，競合が抑えられる．また，競合が発生してしまった場合は，指数バックオフにより一定時間待ってからリトライを行うが，リトライでの成功確率を上げるためにジッター[*17]と呼ばれるリトライ間隔のランダム化を行っている．

また，HBase では，テーブルへの初回もしくは時間があいてからのアクセスに時間がかかることから，参照する可能性のあるテーブルには定期的にバックグラウンドで `Scan` を実行しており，アクセスの少ないテーブルを参照する場合でもレスポンスに時間がかかることがないよう注意している．

Amazon DynamoDB

Amazon DynamoDB[*18]は，Amazon Web Services により提供されている NoSQL データベースサービスである．HBase との機能的な違いとして，DynamoDB では単純な `byte` 列以外にドキュメント型などのデータ型を扱うことができ，データの更新処理についても，`UpdateItem` API を使用して更新内容を DSL の式で記述することで複雑な処理が可能になっている．また，DynamoDB は使用するスループットキャパシティを事前に設定するデータベースである．本システムでは，キャパシティ超過による無駄なリトライを防ぐため，テーブルに設定された書き込みキャパシティを自動で認識して，それを超えないよう書き込みリクエストの送信自体を流量制限している．

本システムのデータモデルと DynamoDB の対応関係は，表 1.3.2 を参照されたい．

[*17] `https://aws.amazon.com/blogs/architecture/exponential-backoff-and-jitter/` が詳しい．
[*18] `https://aws.amazon.com/dynamodb/`

書き込みのバッファリング

イベント処理後のモノイドを毎回ストレージに書き込むのは非効率である．結合則及び交換則が成り立つため，事前に演算 (•) を行うことで，書き込みの回数を減らしストレージへの負荷を軽減することができる．

本システムでは，内部でモノイドの演算 (•) を行う仕組みを加えた遅延キューを利用して書き込みのバッファリングを行っている．この遅延キューの実装では，キューの要素がキーと値を持ち，すでにキュー内に存在するキーと同じキーを持つ要素を追加 (enqueue) した場合，既存の要素との間で演算 (•) が行われ値が更新される．また，一旦キューに追加した要素は，設定した遅延時間を待たなければ取り出すこと (dequeue) が出来なくなっている．ただし，キューには最大容量が設定されており，キュー内の要素数が最大容量の 90% を超えた場合には，遅延時間を待たなくても要素を取り出すことができる．最大容量に達した場合には，追加操作を実行しているスレッドはブロックされる．

3.3　読み込みリクエストの処理

集約結果を取得する読み込みリクエストの処理の流れを図 1.3.3 に示す．高速に結果を応答する必要があるため，キャッシュを行っている．クライアントからのリクエストを受け取ることで処理が始まるが，まず必要なデータを計算し，キャッシュに問い合わせを行い，キャッシュミスしたものについては，Monoid Storage に問い合わせを行いその結果をキャッシュする．最終的には，キャッシュヒットしたもの，Monoid Storage から読み込んだものをマージして，クライアントに返却する．

また，リクエストの実行においては，テーブル毎の同時リクエスト数[*19]を制限している．制限を超えた場合，処理を行わず直ちにエラーが返却されるため，クライアント側ではリトライもしくは，何らかのフォールバック処理を行う必要がある．これは，本システムは複数のサービスから使用されるため，どれか単一のサービスがすべてのリソースを奪うことを防ぎ，何らかの障害などの際に影響範囲を狭めることを目的としている．

読み込みのキャッシュ

利用する用途に応じて要件や負荷の傾向が異なるため，キャッシュを有効にする場合，テーブル毎に以下の項目を設定できるようにしている．

キャッシュ生存期間 (Time to Live)

キャッシュー貫性 書き込み時無効化（`Write Invalidate`）の場合は，ストレージへの書き込み後，対応するキャッシュエントリが無効化される．

キャッシュ対象期間 `INTERVAL` 句による集約期間が過ぎ，一定期間経ったデータのみをキャッシュ対象にする（リアルタイムに変化するデータはキャッシュ対象外にする）．

Redis 上での時系列データのキャッシュ

本システムのデータモデル（図 1.3.1）のデータを効率良くキャッシュすることを考える．ソート済みセットを使用すれば時刻でソートされたままキャッシュ可能だが，キャッシュされていないこととデータがないこと（ネガティブキャッシュ）を効率よく表現することが難しい．そのため，Redis のモジュール機能を利用して，時系列データに適した z-tscache[*20]というデータ型を実装し使用している[*21]．このデータ型では，キャッシュエントリを「時刻範囲 [`time_from`, `time_to`）とその範囲内の時刻から文字列へのソート済みマップ（`map<time, string>`）の組」のソート済みリストとして保持する．時刻範囲が存在するが，マッ

[*19] ある瞬間に処理中のリクエスト数という意味で使用している．秒間リクエスト数とは異なる．

[*20] Redis のユーザデータ型は 9 文字で命名する必要がある．

[*21] ただし，[GROUP BY] INTERVAL 句による期間区切りの集約を行わない場合は，そもそも時系列データとして扱う必要はない．この場合は z-tscache 型を使用せず，組み込みの文字列型を使用している．また，データのシリアライズには Java のシリアライザフレームワークである Kryo (`https://github.com/EsotericSoftware/kryo`) を用いている．

プに対応する時刻がない場合は，データがない（ネガティブキャッシュされている）ものとして扱う．

リスト 1.3.2 に使用例を示す．この例では，はじめに `storeex` コマンドにより，範囲 [2017-12-01, 2017-12-05) のキャッシュをキー `test` に TTL を 300 秒として保存している．次の `load` では，`test` キーに対して範囲 [2017-12-01, 2017-12-31) のキャッシュを問い合わせている．範囲 [2017-12-01, 2017-12-05) はキャッシュヒットとなり，2017-12-02, 2017-12-03 についてはデータが存在し，それ以外はデータが存在しないことを表すネガティブヒットとなっている．結果のない [2017-12-05, 2017-12-31) は，キャッシュミスである．

```
127.0.0.1:6379> tscache.storeex test \
                2017-12-01 2017-12-05 \
                300 # time to live \
                2017-12-02 foo \
                2017-12-03 bar
OK
127.0.0.1:6379> tscache.load test \
                2017-12-01 2017-12-31
1) 1) (integer) 2017-12-01
   2) (integer) 2017-12-05
   3) 1) (integer) 2017-12-02
      2) "foo"
      3) (integer) 2017-12-03
      4) "bar"
127.0.0.1:6379> tscache.invalidate test \
                2017-12-02 2017-12-03
OK
127.0.0.1:6379> tscache.load test \
                2017-12-01 2017-12-31
1) 1) (integer) 2017-12-01
   2) (integer) 2017-12-02
   3) (empty list or set)
2) 1) (integer) 2017-12-03
   2) (integer) 2017-12-05
   3) 1) (integer) 2017-12-03
      2) "bar"
```

ソースコード 1.3.2 z-tscache 型の使用例: 実際は時刻の表現に UNIX 時間 (ms) を使用しているが，読みやすさのためフォーマットしている．

キャッシュサーバーのスケールアウト

Redis は高速ではあるが，1 台では日々増えるトラフィックに対応することはできない．そのため本システムでは，複数のキャッシュノードを起動し，キーのハッシュ値を利用して問い合わせを分散させている．担当するキャッシュサーバーを決める際は，Consistent Hashing を利用している．単純にキーのハッシュ値を mod N（N はキャッシュサーバーの台数）して担当するサーバーを決めた場合，サーバーが増減した際にほとんどのキーの担当サーバーが変わってしてしまい，大量のキャッシュミスが起きてしまう．Consistent Hashing を使用することで，サーバーが増減しても，増えたサーバーもしくは，減ったサーバーが担当していたキーのみ（全体の 1/N）が移動するため，キャッシュミスの発生を低減できる．

また，クラウド環境にデプロイする場合，常に最大のキャパシティに備えるのは経済的ではないため，オートスケーリング機能を使用している．キャッシュノードが動的に変化してしまうため，Apache ZooKeeper や Kubernetes の Endpoints API を用いてキャッシュサーバーの一覧を取得している．

また，Redis は複数の CPU を有効に使えないため，1 ノード上に CPU の物理コア数分，ポートを変えるなどしてプロセスを起動している．

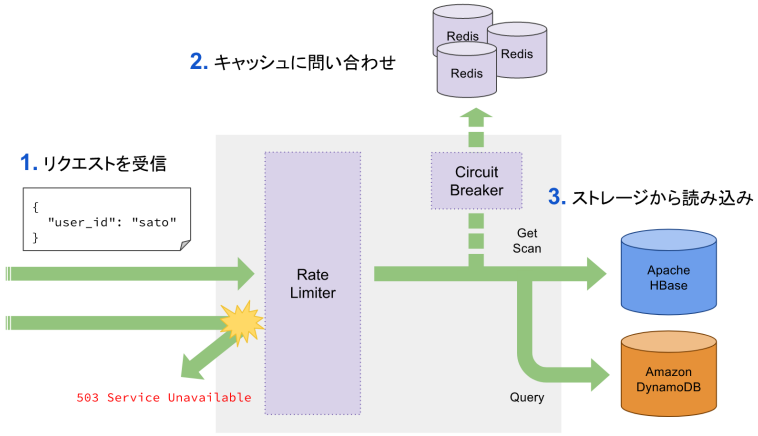


図 1.3.3　リクエストの処理の流れ

4　まとめ

本稿では，ストリーム処理エンジン（開発コード: Zero）を紹介した．「ZeroQL」により宣言的にストリーム処理を記述することで，即座に結果を取得でき，リアルタイムのデータの活用がより短期間で実現できるようになった．

また，広告配信などリクエスト数の多いサービスでも使用されているため，高速にレスポンスを返すための工夫などについても紹介した．

5　今後の展望

今後の予定として，性能などの特性が異なるミドルウェア，例えば，Apache Phoenix, Redis Cluster などへの対応や，JOIN 機能の実装，また，異なり数を計測する集約関数の実装など，機能の充実を図る予定である．また，性能面での改善も継続的に行っていく予定である．

2. 機械学習システム

近年，学術界と産業界の両方で，ユーザ体験の向上を目的としたレコメンデーションシステムの開発に関心が集まっている．例えば，世界最大のインターネットオークションサイトを運営する eBay は，社内で開発したレコメンデーションシステムの学術報告に取り組んでおり，新しいアルゴリズムやシステムにより CTR の向上などサービス改善に寄与していることが報告されている．また，動画ストリーミング配信サービスを展開する Netflix は，Netflix Prize と呼ばれるコンペティションを開催し，自社のレコメンデーションシステムを最も改善するアルゴリズムを提案した参加者に賞金を授与している．このように，様々な企業がサービスの競争力を高める上でレコメンデーションシステムの開発を重要視しており，積極的に投資を行っている．

そうした状況下で，秋葉原ラボにおいても，Ameba ブログ・AbemaTV・AWA や広告プロダクトなどから日々生成される大量のデータを活用したレコメンデーションシステムを開発し，サービス改善に貢献している．当社は数多くのサービスを運営しており，中には Ameba ブログのように国内有数の規模を誇るものもある．これらの要件に対応するために，秋葉原ラボではレコメンデーションシステムについて次の課題に取り組んでいる．

第一に，サービスからの多種多様な要望に対する汎用的なレコメンデーションシステムの開発である．サービスごとのデータソースの取得方法や類似したロジックを共通化することで，運用・管理コストや新たなサービスへの導入コストが改善できる．加えて A/B テストやバンディットアルゴリズムを用いたコンテンツのリランキング，CTR など各種指標のレポーティングの機能などを提供することで，サービスが抱える様々な要求を満たしている．

第二に，大規模データに耐えうるスケーラブルなレコメンデーションシステムの実現である．レコメンデーションシステムが扱うデータ量はサービスの成長に伴って増加するため，システムのスケーラビリティは必須である．また，大規模データに対して高速なレコメンドを実現するためにはアルゴリズムの計算量やメモリ効率についても最適化が求められる．

2.1 章では，秋葉原ラボが開発するレコメンデーション基盤のシステムアーキテクチャについて解説する．2.2 章では，A.J.A. Recommend Engine で用いている高速な探索アルゴリズムについて紹介する．

2.1 大規模リアルタイムレコメンデーションを支えるスケーラブルなシステム基盤の構築

Juhani Connolly, 内藤 遥, 福田 鉄也

概　要　レコメンデーションシステムを開発する上で，最新アイテムのオンラインでの取り込みやモデルの定期更新，CPU やメモリなどのリソース管理は重要なファクターとなっている．また数多くのメディアサービスを抱える大規模なシステムを考えた場合，新規メディアサービスやデータセットの増加に応じてシステムが自動でスケールできる状態が望ましい．本稿では上記を解決するためのシステムアーキテクチャ，およびシステムの様々な処理を効率化する各コンポーネントについて紹介する．

Keywords　レコメンデーションシステム, スケーラビリティ, リソース管理

1　背景

秋葉原ラボでは様々なメディアサービスに対してレコメンデーションシステムを提供している．レコメンデーションシステムが扱うデータ量はシステムを利用するメディアサービスの成長に伴って増加するため，データ規模に対するスケーラビリティが必要とされる．特に当社の文書推薦システムは数百のメディアサービスに対し推薦を提供しており，そのために柔軟にスケールできるように設計されている．

文書推薦システムは文書に対してそれと類似している文書をメディアサービス内から検索し，推薦結果として提供するものである．本システムは一般的な類似文書探索と同様に，文書のテキストデータをベクトル化して次元圧縮を行い，それを基に文書間の類似度を計算している．ラボが従来提供していた文書推薦システムはキャッシュ効率などの観点から，メディアサービスごとにサーバを設置してベクトル化や次元圧縮の処理を行っていた．しかし，システムを利用するメディアサービスの増加に伴って，サーバ維持コストや新規メディアサービス導入にあたる運用コストが問題となった．これらの問題を解決するために，本システムは頻繁に行われるメディアサービスの追加におけるデータセットの更新の自動化, およびリソース管理の効率化を目指して開発されている．具体的には，次のような要件を定めた．

- 新規メディアサービスの導入コスト削減
- 文書の追加，更新，削除時におけるリアルタイムなデータセットの更新
- 新規ワード抽出のための定期的なモデルの更新，およびダウンタイムが生じないシステムへの反映
- 一部のサーバがダウンしてもフェイルオーバーできる冗長化構成

本章ではこれらの要件を満たすレコメンデーションシステムのアーキテクチャについて紹介する．

2 システム構成

本システムは複数のコンポーネントで構成されている．主なコンポーネントとその役割について次に示す．

API 文書ベクトルを保持し，クエリに対して類似文書を返す窓口

Diffstore 各メディアサービスのデータセットをリアルタイムで更新するシステム

Model Manager メディアサービス追加時などのリソースの自動割り当てや，API 群のモデル更新のタスク管理を行うシステム

ZooKeeper API など分散システムのコーディネートを支援するサービス[*1]

スケールアウトを自動化するために AWS[*2]のオートスケーリング機能を使っており，またサーバ間のメッセージングには Avro[*3]を利用している．システム構成の全体図を**図 2.1.1** に示す．

2.1 API

API はメディアサービスからのリクエストを処理し，推薦結果を提供するコンポーネントである．推薦結果を作成するため文書のベクトル表現を保持しており，近傍探索によって推薦結果となる文書を選択する．

本システムではベクトル量子化を採用し，圧縮した文書のベクトル表現をすべてメモリ上に展開している．量子化によって 1 文書あたり数 10 バイトで文書ベクトルを表現することで，効率的にメモリ空間を利用している．詳細については「直積量子化に基づく高速な類似文書の探索」の章を参照されたい．

量子化されたベクトル間の類似度を比較するためには，量子化したベクトルとコードブックが必要である．一方，ベクトル化や次元圧縮の過程で発生する中間生成物は類似度の比較には不要である．従来のシステムでは新しい文書の追加に備えて中間生成物を API サーバ上で保持していたが，本システムでは API サーバは類似度計算に必要なデータのみを保持し，ベクトル化や次元圧縮の処理を Diffstore に分離した．定期的に Diffstore 上のデータセットと同期をとることで API サーバの保持するデータを最新に保っている．

また，冗長化および負荷分散のため，1 つのメディアサービスに対して複数の API サーバが推薦を提供している．各 API サーバが担当するメディアサービスはリソースの空き状況などを元に Model Manager によって決定される．

2.2 Diffstore

直積量子化によって文書をベクトルに変換する際，特異値分解による次元圧縮が行われるが，このためには巨大な行列をメモリ上に展開する必要がある．旧システムではモデルはデータセットとともに API サーバに保存されており，これによって以下に挙げる問題が生じていた．

- データセット以上にモデルのデータ量が増大する場合があり，その場合メモリが圧迫される
- 各 API サーバでデータセットの追加や更新を行っており，処理が非効率である
- API サーバ間でデータの整合性を保つことが困難で，データセットの更新に失敗した際のリカバリなど含め API は互いの状態を把握する必要がある

これらの問題を解決するために開発されたのが Diffstore である．Diffstore では各メディアサービスのモデルと，直近で更新されたデータセットのみを持つ．前回の取得から更新のあったデータセットを定期的にダウンロードし，各文書のベクトル化，および量子化を行う．それぞれのデータはデータベースへの書き出しに加えて一時的にメモリ上に保存され，各 API サーバからのポーリン

[*1] https://zookeeper.apache.org/
[*2] https://aws.amazon.com/jp/
[*3] https://avro.apache.org/

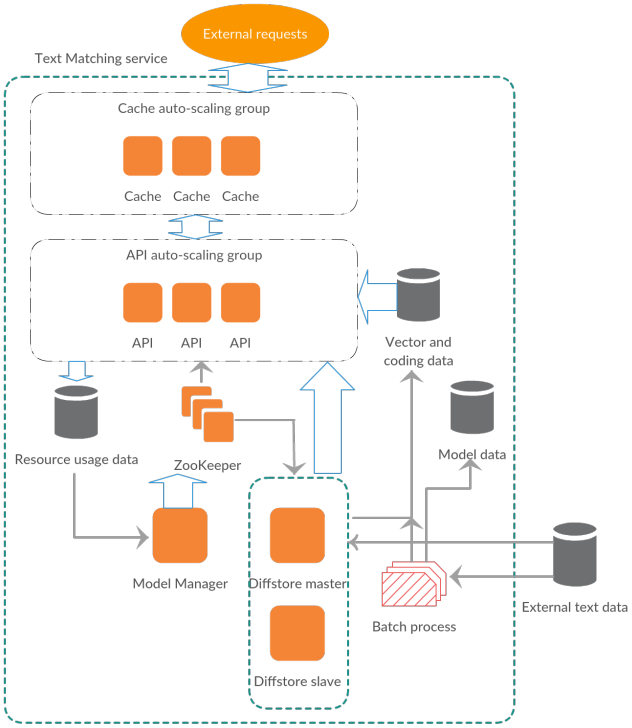


図 2.1.1　アーキテクチャの全体図．詳細は一部省略している．Batch process は定期的にメディアサービスのモデルを生成するコンポーネント，Cache はクライアントと API 間のキャッシュを行うコンポーネントである．

グを受けて転送される．このアプローチによって処理がモジュール化され，API サーバの役割を類似文書の探索に限定でき，容易にスケールできるようになった．

また，Diffstore はメモリ使用量削減のため，ローカルディスクに保存してあるデータを必要に応じてメモリに展開している．頻繁にデータセットが更新されるメディアサービスのモデルはメモリ上に残り，逆に更新頻度の少ないメディアサービスのモデルはディスクに戻される．これによって，メモリ空間が効率的に利用できるようになった．

ところで，Diffstore がダウンした場合，最新の文書はデータセットに追加されなくなる．ブログやニュースなどのメディアサービスでは，最新の記事がもっともアクセスされやすい傾向にあるため，新規文書が推薦結果に反映されないことは大きな問題となりうる．このため Diffstore はマスター・スレーブ構成で冗長化されている．現状はこの構成で充分だが，将来的にメディアサービスが増えリソース不足に陥る場合に備え，API と同様のメディアサービスの自動割り当てを導入することを検討している．

2.3　Model Manager

背景で述べたように，本システムでは数百のメディアサービスが動く想定でスケールを考える必要がある．全ての API サーバが全メディアサービスのベクトルデータを保持するとメモリなどのリソース効率が悪いので，メディアサービスごとのリソース使用量に応じて適切な API サーバの数が割り当てられている状態が望ましい．さらにはその割り当ての自動化までできていると，運用での負担が軽減される．Model Manager は上記の要望を元に開発された API のリソース管理，およびメディアサービス割り当てのスケジューリングを行うシステムである．Model Manager は API サーバのアプリケーションから各 API サーバ単位，もしくは API サーバ内のメディアサービス単位でリソース使用状況をリアルタイムで収集しており，

これを基にメディアサービスの割り当てを決定している．

API サーバのメディアサービスへの割り当ては 3 つの構成要素で実現されている．それぞれの役割について下記に示す．

Trigger 割り当ての発動条件を表す．例えば「メディアサービスが追加されたとき」「メディアサービスの CPU 使用率が閾値より高いとき」などのような設定が可能である．

Action 割り当ての内容を表す．例えば「トリガ対象のメディアサービスを CPU 使用率の低い API サーバに追加する」などのようなアクションを設定する．

Restriction Action の制限事項を表す．API サーバのトータルメモリ使用量の上限や，メディアサービスに割り当てられた API サーバ数の下限などを設定できる．

これらの構成要素はそれぞれ管理画面上で設定可能であり，柔軟かつ強固なリソース管理を実現している．

2.4 ZooKeeper

ZooKeeper は分散システムのためのコーディネーションサービスであり，階層的な名前空間へのデータ保存や書き込み結果の通知機能などを有している．本システムでは各 API のメディアサービスへの割り当てやモデル更新の際に発生するタスクの制御で ZooKeeper を利用している．ZooKeeper ではタスク管理用の `assigned` ノード，モデル管理用の `serving` ノードをそれぞれ保持しており，`assigned` ノード配下に追加したモデル ID を持つモデルの更新が完了すると，`assigned` ノード配下から `serving` ノード配下へモデル ID が移動する仕組みとなっている．モデル更新のタスク制御フローを**図 2.1.2** に示す．各処理の内容は次の通りである．

1. `assigned` ノード配下の旧モデル ID を削除
2. API に通知
3. API 上から旧モデルおよびベクトルデータを削除
4. `serving` ノード配下の旧モデル ID を削除
5. Model Manager に通知
6. `assigned` ノード配下に新規モデル ID を登録
7. API に通知
8. DB から新規モデルおよびベクトルデータを取得
9. `serving` ノード配下に新規モデル ID を登録

3 オートスケーリング

メディアサービスが頻繁に追加された場合，特にデータセットが大きいメディアサービスではリソース管理やそれに伴うサーバの増設をマニュアルで行うのは困難であり，運用コストがかかる．以上の内容を考慮して AWS のオートスケーリング機能を API とキャッシュ層で使用している．

リソースの使用量が閾値を超えた際に，サーバの増設からアプリケーションのデプロイ・起動までが必要に応じて自動で行われる．新たに起動されたサーバは Model Manager によって直ちにディスカバリされる．

4 今後の課題

本章で紹介したシステムの一部はマスターとなるノードのリソース不足がボトルネックになりうる．API のノード数はクラウドプロバイダの限界までスケールできるが，現状ではリソースの割り当てを担う Model Manager のスケーラビリティには限界があるため，さらなる高負荷が予想される場合は分散処理構成に変える必要がある．

同様に Diffstore はモデルをローカルのメモリやディスク上に持つ必要があり，冗長化はされているがスケールできない現状にある．データ量の大きなモデルはディスク上に書き込んでおき，必要に応じてメモリに昇格させているが，メディアサービスが増えるとディスク I/O の限界に陥る可能性があるため，長期的には Diffstore も分散させ

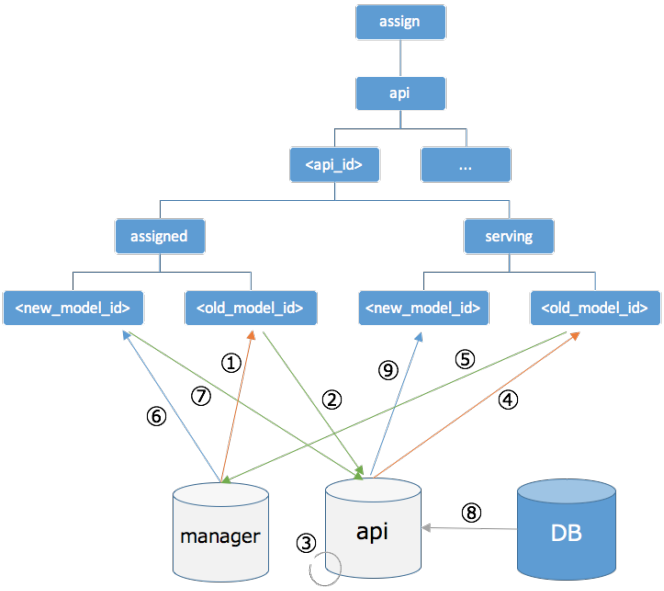


図 2.1.2　ZooKeeper によるモデル更新のタスク制御

る必要がある．

また，本システムはバックエンドでベクトルデータを持つデータベースに強く依存しており，DB の負荷高騰などに起因して全体のレスポンスも遅れてしまう点も課題である．対策としてはデータベースを介さず量子化ベクトル同士の類似度を計算することが考えられるが，文書の近傍探索の精度は下がってしまうため現状では採用していない．

5　まとめ

本章では数百のメディアサービスが利用可能なレコメンデーションシステムの構成を紹介した．ベクトル量子化によるデータ圧縮と近傍探索の高速化を始め，Diffstore による差分更新データの処理の分離や Model Manager によるリソース管理によって旧システムの非効率な処理を改善できた．またオートスケーリングを使って各コンポーネントの冗長化をしており，耐障害性を担保している．これらの仕組みによって新規メディアサービスの導入コストも改善でき，安定したシステムを提供する事が可能になった．

2.2 直積量子化に基づく高速な類似文書の探索

福田 鉄也，Juhani Connolly，内藤 遥

概　要　本稿では，AJA Recommend Engine における類似文書の探索アルゴリズムについて紹介する．本システムが扱う問題では，探索対象の文書は頻繁に追加，または削除されるため，ある文書に対する類似した文書のリストはリクエストの都度計算される必要がある．一方で，数百のメディアサービスに対してそれぞれ数万から数十万の文書を保持する必要があるため，リクエストの都度すべての文書を素朴に走査することは計算量などの観点から現実的ではない．そこで我々のシステムでは，文書の tf-idf を特異値分解によって低次元の密なベクトルとして表現した後，直積量子化と呼ばれる手法で省メモリかつ高速に近似探索できる形に変換して保持することで，これらの問題に対応している．また，実サービスのデータセットに対して適用した際の精度について確認し，その結果について述べる．

Keywords　類似探索，次元圧縮，ベクトル量子化

1　はじめに

AJA Recommend Engine[*1]は株式会社 AJA が提供する推薦システムのパッケージである．同レコメンドエンジンはニュースサイトやウェブマガジンのようなメディアサービス向けに提供されており，各メディアサービスにおける記事コンテンツの推薦などを実現している．本稿で紹介するシステムは AJA Recommend Engine のコンポーネントとして開発された, 文書の類似度に基づいた推薦システムである．このシステムはメディアサービスの特性から次のような要件を満たすものとして設計されている．

1. 単一のシステムで数百のメディアサービスまで対応できる拡張性を持つこと
2. 1 つのメディアサービスあたり，数万から数十万件の文書を保持できること
3. 文書の頻繁な追加と削除に対応できること

類似文書の探索問題は，それぞれの文書を何らかの方法でベクトルとして表現した後，ベクトル間の「類似度」の上位を返却する形で扱われることが多い．我々のシステムにおいて，文書は頻繁に追加，あるいは削除されるため，返却する文書のリストはリクエストの都度計算される必要があるが，数万のベクトルを素朴にすべて走査することは計算量の観点から現実的ではない．また，高速に結果を計算するためベクトルの情報はすべてメモリ上に保持したい一方で，256 次元のベクトルの各要素を 64 ビットの倍精度浮動小数点数の形で扱った場合 10 万件で 200MB 程度となるため，これを数百のメディアサービスに対して保持することはメモリ消費の面からも効率的ではない．

そこで我々のシステムでは，文書の tf-idf を特異値分解によって低次元の密なベクトルとして表現した後，直積量子化と呼ばれる手法で省メモリかつ高速に近似探索できる形に変換して保持する

[*1] https://aja-kk.co.jp/service/aja-recommend-engine

ことで，これらの問題に対応している．

以降，2 節では，文書をベクトルとして表現することで類似した文書を探索するための枠組みについて述べ，3 節では，直積量子化に基づくベクトル空間上で類似したペアを高速に近似探索するためのアルゴリズムについて紹介する．また，4 節では，直積量子化を実際のサービスのデータセットに対して適用した場合の精度について確認する．

なお，これらのアルゴリズムを活用した実際のシステム構築については「大規模リアルタイムレコメンデーションを支えるスケーラブルなシステム基盤の構築」の章を参照されたい．

2　ベクトル空間モデルに基づく文書表現

2.1　ベクトル空間モデル

例えば，ある用語が文書の中で繰り返し出現しているとき，その文書はその用語に強く関連していると考えられる．また，複数の文書に出現する用語よりも，一部の文書のみに出現する用語の方が文書を特徴付ける上で重要であると考えられる．これらを表現するため，**tf-idf** と呼ばれる各用語の各文書に対する複合的な重みを考える．tf-idf は，用語 i，文書 j に対して次式で与えられる．

$$\text{tf-idf}_{i,j} = \text{tf}_{i,j} \times \text{idf}_i \tag{2.2.1}$$

$$\text{tf}_{i,j} = \frac{n_{i,j}}{\sum_{i' \in i} n_{i',j}} \tag{2.2.2}$$

$$\text{idf}_i = \log \frac{n}{\text{df}_i} \tag{2.2.3}$$

ただし，$n_{i,j}$ はある用語 i の文書 j 内での出現回数，n は全文書数，df_i は用語 i が出現する文書の数とする．

これにより，各文書を式 (2.2.1) で与えられる要素を持ったベクトルと見なすことができる．このように，文書集合を共通のベクトル空間においてベクトルとして表現する方法は**ベクトル空間モデル**として知られる．2 つの文書の類似度は，それぞれのベクトル表現の間の類似度によって与えられる．類似度としては，**コサイン類似度**などが利用されることが多い．

2.2　特異値分解による次元圧縮

前節で述べたベクトル空間表現では，全ての用語は全く異なる意味を持っているとして扱われる．しかし，たとえば同義語や類義語のような，異なる意味として扱うべきではない用語も存在する．ここでは，同義語や類義語を考慮したより適切なベクトル表現を得る方法について説明する．

用語数を m として，各列に式 (2.2.1) によって与えられる文書ベクトルを持つ $m \times n$ の行列を考える．いま，任意の行列 $\mathbf{C} \in \mathbb{R}^{m \times n}$ に対する**特異値分解**が次の形で存在する．

$$\mathbf{C} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top \tag{2.2.4}$$

ただし，$\mathbf{U}$ は $m \times m$，$\mathbf{V}$ は $n \times n$ の直交行列であり，$\mathbf{\Sigma}$ は $m \times n$ の対角行列であるとする．また，$\mathbf{\Sigma}$ の対角成分は大きい順に並んでいるとする．これらの対角成分は特異値と呼ばれる．

ここで，$\mathbf{\Sigma}$ から大きい順に d 個の特異値のみを残した $d \times d$ の対角行列 $\tilde{\mathbf{\Sigma}}$ と，$\mathbf{U}$ と $\mathbf{V}$ から $\tilde{\mathbf{\Sigma}}$ に対応する部分以外を取り除いた行列 $\tilde{\mathbf{U}} \in \mathbb{R}^{m \times d}$，$\tilde{\mathbf{V}} \in \mathbb{R}^{n \times d}$ を考える．再構成行列 $\tilde{\mathbf{C}} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^\top$ はランク d の行列のうち，$\mathbf{C}$ との差分行列のフロベニウスノルムが最小のものになっている．

文書行列 $\mathbf{C}$ に特異値分解を適用した結果を考えると，$\tilde{\mathbf{V}}$ の各行は，それぞれ対応する文書を d 次元の空間で表現したものといえる．$\tilde{\mathbf{C}}$ は $\mathbf{C}$ に対して良い近似となっていることから，それぞれの文書間のコサイン類似度の相対的な値は保たれていることが期待できる．加えて，用語と文書をそれぞれ d 次元の空間で表現するにあたって，似たような共起を持つ用語が 1 つの次元として表現されることで，前述の同義語の問題に対応できることが期待される．

なお，特異値分解の計算コストは著しく大きいため，我々のシステムでは乱択アルゴリズムに基づく方法 [2] を採用している．

3 直積量子化による近傍探索の高速・省メモリ化

2 節では文書をベクトルとして表現する手法について説明した．ここで，類似文書の探索は，あるベクトルに対してコサイン類似度が最大となるベクトルを探索する問題へと置き換えて考えることができる．また，すべてのベクトルが単位ベクトルに正規化されているとき，ユークリッド距離を最小にするベクトルと，コサイン類似度を最大にするベクトルは同一のものとなっている．すなわち，ベクトルの集合 $\mathcal{Y} \subset \mathbb{R}^d, |\mathcal{Y}| = n$ から，クエリベクトル $\mathbf{x} \in \mathbb{R}^d$ とのユークリッド距離が最小となるベクトル $\mathbf{y}^*$ を見つけたい．

$$\mathbf{y}^* = \arg\min_{\mathbf{y} \in \mathcal{Y}} d(\mathbf{x}, \mathbf{y}) \tag{2.2.5}$$

ただし $d(\cdot, \cdot)$ はユークリッド距離である．

式 (2.2.5) を総当りで計算した場合の計算量は $O(nd)$ であり，n が大きいとき現実的ではない．また，$\mathcal{Y}$ の空間計算量も $O(nd)$ となり，メモリ上に保持するのは難しい．

3.1 ベクトル量子化

ベクトル量子化は，ベクトルを圧縮して表現することで空間計算量の削減を図る手法である．量子化器 q は，ベクトル $\mathbf{y} \in \mathbb{R}^d$ をベクトル $q(\mathbf{y}) \in \mathcal{C} = \{\mathbf{c}_i; i \in \mathcal{I}\}$ に写像する関数である．ただし，インデックスの集合 $\mathcal{I} = 1, \ldots, C$ は有限であるとする．$\mathbf{c}_i$ を**セントロイド**と呼ぶ．また，その集合 $\mathcal{C}$ を**コードブック**と呼ぶ．コードブック $\mathcal{C}$ および関数 q は次式で定義される量子化誤差 E を最小化するものとして計算される．

$$E = \frac{1}{n} \sum_{\mathbf{y} \in \mathcal{Y}} d(\mathbf{y}, q(\mathbf{y}))^2 \tag{2.2.6}$$

これは k-means アルゴリズムによって局所解の 1 つを求めることができる．

3.2 直積量子化

ベクトル量子化によって 128 次元のベクトルを 64 ビット，すなわち，1 次元あたり 0.5 ビットで表現することを考えよう．このときセントロイドの数は $C = 2^{64}$ となり，最適化に C の数倍の学習データを必要とする k-means アルゴリズムを適用することは現実的ではない．**直積量子化** [4] はこのような問題に対応するための方法の 1 つである．直積量子化では，入力ベクトル $\mathbf{y}$ を K 個の部分ベクトルに分割することを考える．

$$\mathbf{y} = [\mathbf{y}^{(1)}; \mathbf{y}^{(2)}; \cdots; \mathbf{y}^{(K)}] \tag{2.2.7}$$

ただし，$\mathbf{y}^{(k)} \in \mathbb{R}^{d/K}$ とする．すべてのベクトルの k 番目のブロックを含む部分空間は，コードブック $\mathcal{C}^{(k)}$ によって量子化される．コードブック全体はその直積 $\mathcal{C} = \mathcal{C}^{(1)} \times \cdots \times \mathcal{C}^{(K)}$ によって表される．いまコードブック $\mathcal{C}^{(k)}$ が任意の k について C^* のセントロイドを持つとき，直積量子化によって $C = (C^*)^K$ 個のセントロイドを表現できたといえる．

直積量子化の利点は，コードブック全体を複数の小さなコードブックから構成していることである．小さなコードブックにおけるセントロイド数は少数で良いので，現実的な学習データ量で最適化することができる．また，各コードブックのセントロイド数 $C^* = 256$ のとき，これは 8 ビットで表現できる．ベクトルの分割数 $K = 16$ であれば 1 つのベクトルあたり 128 ビットで表現できるので，各セントロイドを保持することによるオーバーヘッドを考慮しても良いメモリ効率を得られる．

3.3 探索

クエリベクトル $\mathbf{x}$ とのユークリッド距離が最小となるベクトルを $\mathcal{Y}$ から近似探索する方法について，$\mathbf{x}$ の量子化を行う方法と，行わないものの 2 つが考えられる．ベクトル $\mathbf{x}$ と $\mathbf{y} \in \mathcal{Y}$ のいずれも量子化を行うとき，2 つのベクトル間の距離は次式で近似される．

$$d(\mathbf{x}, \mathbf{y}) \approx \sqrt{\sum_k d(q^{(k)}(\mathbf{x}^{(k)}), q^{(k)}(\mathbf{y}^{(k)}))^2} \tag{2.2.8}$$

ただし $q^{(k)}$ は k 番目の部分空間における量子化器である．すべてのデータベースベクトル $\mathbf{y}$ に対し

てユークリッド距離を近似計算するときの計算量は，セントロイド間の距離を事前計算すれば $\mathbf{x}$ の量子化について $O(C^*d)$，各ベクトルに対する距離の計算について $O(nK)$ となる．

一方，データベースベクトル $\mathbf{y}$ については量子化するが，クエリベクトル $\mathbf{x}$ は量子化しない場合，2 つのベクトル間の距離は次式で近似される．

$$d(\mathbf{x},\mathbf{y}) \approx \sqrt{\sum_k d(\mathbf{x}^{(k)}, q^{(k)}(\mathbf{y}^{(k)}))^2} \qquad (2.2.9)$$

この方法ですべてのデータベースベクトルに対してユークリッド距離を近似計算すると，先述の方法に比べて近似精度は良くなる一方, 探索時のメモリ消費量は大きくなる．計算量については $\mathbf{x}$ と全てのセントロイドの距離の計算について $O(C^*d)$, 各ベクトルに対する距離の計算について $O(nK)$ となり，先述の方法と等しい．

3.4　粗い量子化による高速化

3.3 項で述べた方法は総当りの計算と比較して高速であるが，依然として計算量は n に依存している．そこで，事前に探索空間をパーティションに分割することで n に依存した探索時間を避けることを考える．ここでは，探索空間の分割にもベクトル量子化を利用する．まず，セントロイド数が $1,000$ から $1,000,000$ 程度の**粗量子化器** q_c による**粗い量子化**を行う．その後，粗い量子化によって得られたセントロイドとの残差ベクトル $r(\mathbf{y}) = \mathbf{y} - q_c(\mathbf{y})$ に対して直積量子化を実行する．つまり，データベースベクトル $\mathbf{y}$ は次式によって近似される．

$$\mathbf{y} \approx q_c(\mathbf{y}) + q(\mathbf{y} - q_c(\mathbf{y})) \qquad (2.2.10)$$

また，クエリベクトル $\mathbf{x}$ と $\mathbf{y}$ の距離は次式によって近似計算される．

$$\begin{aligned} d(\mathbf{x},\mathbf{y}) &\approx d(\mathbf{x} - q_c(\mathbf{y}), q(\mathbf{y} - q_c(\mathbf{y}))) \\ &= \sqrt{\sum_k d((\mathbf{x} - q_c(\mathbf{y}))^{(k)}, q^{(k)}((\mathbf{y} - q_c(\mathbf{y}))^{(k)}))^2} \end{aligned} \qquad (2.2.11)$$

クエリベクトル $\mathbf{x}$ の近似近傍探索は，$q_c(\mathbf{x})$ 近傍のいくつかのパーティションに割り当てられたデータベースベクトルのみを走査することによって実行される．

3.5　ベクトル分割の最適化

最小のユークリッド距離を持つベクトルを探索する問題は，入力空間の移動，回転に対して不変であるので，事前に入力空間を直交行列 $\mathbf{R}$ によって変換することで，より量子化誤差を小さくするようなベクトル分割を得られることが期待できる [1]. 具体的には，下記のような最適化問題を考える．

$$\begin{aligned} &\min_{\mathbf{R},\mathcal{C}^{(1)},\ldots,\mathcal{C}^{(K)}} \sum_{\mathbf{y}\in\mathcal{Y}} d(\mathbf{y}, q(\mathbf{y}))^2 \\ &\text{s.t.} \quad q \in \mathcal{C} = \{\mathbf{c} | \mathbf{Rc} \in \mathcal{C}^{(1)} \times \cdots \times \mathcal{C}^{(K)}, \mathbf{R}^\top\mathbf{R} = \mathbf{I}\} \end{aligned} \qquad (2.2.12)$$

紙幅の都合上，詳細は省略するが $\mathbf{R}$ と $\mathcal{C}^{(1)},\ldots,\mathcal{C}^{(K)}$ を交互に更新することで局所解の 1 つを求めることができる．

4　精度評価

実サービスのデータセットにおけるパラメタ設定，およびベクトル分割の最適化の影響を確認するため，簡単な実験を行った．対象データとして，Spotlight[*2]からランダムに抽出した 10,000 件の文書データを用いた．アルゴリズム `PQ` が直積量子化によるもの，`OPQ` が直交行列 $\mathbf{R}$ によってベクトル分割を最適化した直積量子化である．クエリベクトルとの距離計算は，クエリベクトルの量子化を行わない方法によって計算した．また，いずれの場合も特異値分解後のベクトルの次元数は 256 に設定した．各設定における Recall@10 および Recall@100 を表 2.2.1 に示す．C^*, K が上がるにつれて近似精度が高まり，Recall@10, Recall@100 が改善していることを確認できる．また，ベクトル分割の最適化によってもそれぞれの値が改善していることを確認できる．

[*2] `http://spotlight-media.jp/`

表 2.2.1 各設定における MSE と Recall

アルゴリズム	C^*	K	Recall@10	Recall@100
PQ	256	16	0.356	0.709
PQ	1024	16	0.439	0.777
PQ	256	32	0.673	0.943
PQ	1024	32	0.790	0.979
PQ	256	64	0.933	0.999
PQ	1024	64	0.982	1.000
OPQ	256	16	0.478	0.833
OPQ	1024	16	0.549	0.861
OPQ	256	32	0.747	0.966
OPQ	1024	32	0.814	0.987
OPQ	256	64	0.952	0.999
OPQ	1024	64	0.985	1.000

5 まとめ

本稿では，文書をベクトルとして表現することで類似した文書を探索するための枠組みと，ベクトル空間上で類似したペアを高速に近似探索するための直積量子化と呼ばれるアルゴリズムについて紹介した．なお，文書を低次元のベクトル空間で表現するための，より精緻な方法 [3, 5] の採用は今後の課題である．

参考文献

[1] Ge, T. et al.: Optimized product quantization, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.36, No.4, pp.744–755 (2014).

[2] Halko, N., Martinsson, P.G. and Tropp, J.A.: Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, *SIAM review*, Vol.53, No.2, pp. 217–288 (2011).

[3] Hill, F., Cho, K. and Korhonen, A.: Learning Distributed Representations of Sentences from Unlabelled Data, *Proc. of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp.1367–1377 (2016).

[4] Jégou, H., Douze, M. and Schmid, C.: Product quantization for nearest neighbor search, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.33, No.1, pp.117–128 (2011).

[5] Kenter, T., Borisov A. and Rijke M.: Siamese CBOW: Optimizing Word Embeddings for Sentence Representations, *Proc. of the 54th Meeting of the Association for Computational Linguistics*, pp.941–951 (2016).

3. データ分析

近年，Google や Yahoo!を初めとした Web サービスを運営している企業は膨大な数のユーザの行動をログデータとして収集し分析することで，サービスの改善やユーザターゲティングなどへの応用を進めている．また，Facebook などのソーシャルネットワーキングサービスでは，データをサービスの運営に役立てるだけでなく，サービスの使われ方を社会科学の観点から分析する研究も行われている．

当社はアメーバブログ・ピグパーティをはじめとした Web サービスに加えて，ソーシャルゲームや AbemaTV・AWA などの数多くのメディアサービスを開発・運用している．ユーザ行動の性質や解決すべき課題はサービス毎に異なるため，それぞれに応じた分析を行う必要がある．我々は日々生成される大量のログデータの記述統計的な分析に加えて，解決すべき課題に応じた統計的モデリングによる行動分析を行っている．これらの研究開発を通して，ユーザ行動を理解しサービスの改善や発展に貢献するとともに，ヒトや社会の理解に寄与し Web サービスの活発な利用を促進する研究開発に取り組んでいる．

3.1 章では，インターネットテレビサービス AbemaTV におけるユーザの視聴行動のログについて，MARS モデルを適用してユーザのアクティブ度と継続離脱分析を行った事例について紹介する．

3.2 章では，音楽ストリーミングサービス AWA におけるユーザの聴取行動のログについて，隠れマルコフモデルを適用して聴取トレンドの変化分析を行った事例について紹介する．

3.3 章では，Web サービスが生み出すビッグデータを用いてヒトや社会の理解にアプローチする社会科学の研究事例として，ソーシャルゲーム上での協調行動の分析および社会構造のモデル化と，ピグパーティにおけるソーシャルサポートの研究について紹介する．

3.1 インターネットテレビサービス AbemaTV におけるユーザアクティビティ分析

和田 計也

概　要　本稿では，株式会社 AbemaTV がインターネット上で展開している AbemaTV サービスを例にとり，サービスの継続・離脱分析を週単位で行う方法および，同時にユーザのアクティブ度を定量化する方法を提案する．具体的には分析モデルとして multivariate adaptive regression splines（MARS）を用い，翌週アクティブ化日数を目的変数とした二項分布を仮定することで翌週単位での継続・離脱分析を実現しており，二項分布での成功確率 p をユーザのアクティブ度と定義することで定量化を実現している．また実ビジネス上のデータでよく直面する，正規分布ではない分布のデータであっても MARS を用いることでうまくモデル化できることが示された．

Keywords　離脱分析，MARS，インターネットテレビ

1　はじめに

インターネットの発展に伴い，インターネット上で展開されるサービスは従来の単純な Web サイトから Facebook[*1]や Twitter[*2]等の SNS に至るまでたくさんの種類のサービスが出現してきた．最近では Netflix[*3]や Hulu[*4], YouTube[*5]などインターネット上で動画を視聴できるサービスも展開されており，80% 以上の人が利用したことがあると答えるほど一般的になってきている [1]．株式会社 AbemaTV でもインターネットテレビサービスである AbemaTV サービス[*6]を展開している．これはインターネット上で，オリジナルの生放送コンテンツや，ニュース，音楽，スポーツ，アニメなど多彩な番組が楽しめる約 20 チャンネルをすべて無料で視聴することが可能なサービスである．このようなインターネット上のサービスは据置型のゲームやテレビと比べて，ユーザにとっては初期費用がかからないなどの理由により導入が簡単な反面，サービス利用継続率が時間とともに低下していきやすいという問題を抱えている [2]．そのためユーザのサービス内での行動を分析することで離脱しないような要因を発見してサービス改善に役立てることができれば，ユーザにとってもサービス運営側にとってもお互い良いことだといえよう．本稿では AbemaTV サービスにおけるユーザのアクティブ度定量化およびユーザの継続・離脱行動を分析した事例を紹介する [3]．

本研究においてサービス側から求められた要件は以下の 2 件であった．

1. 人が見て解釈可能であること

[*1] https://www.facebook.com
[*2] https://twitter.com
[*3] https://www.netflix.com
[*4] https://www.happyon.jp
[*5] https://www.youtube.com
[*6] https://abema.tv

2. 変数選択ができること

またその他に，AbemaTV の視聴データは正規分布に従わないため以下の点も考慮する必要があった．

1. 正規分布を仮定しないこと

これらの要件を満たす手法として multivariate adaptive regression splines（MARS）[4] を本研究では用いた．また，AbemaTV はリニア配信でかつ 1 週間単位の番組表に従ってコンテンツを提供するサービスであることから，翌週のアクティブ日数（≒サービス利用日数）を目的変数とした継続・離脱分析を行った．

2　先行研究

片岡らは POS データに Classification by aggregating emerging pattern（CAEP）を適用させて優良顧客の離脱予測モデルを構築した [5]．また，佐藤らはオンラインソーシャルゲームの行動ログを混合正規分布によるクラスタリングにより，ユーザの離脱傾向を研究した [6]．藤井らは決定木を用いて，CD 購買 POS データから継続購入するユーザのモデリングを行った [7]．このように，さまざまなデータに対して分類器を用いてユーザの離脱分析を行う事例は従来から行われている．Chou らは MARS モデルで変数選択を行いニューラルネットワークで予測モデルを構築する手法を用いて，乳がんの術後再発症パターンを腫瘍サイズ等の所見データから作成した変数を用いてマイニングを行った [8]．このように分類器に MARS モデルを適用する事例も行われている．

3　方法

3.1　利用したデータおよびデータ収集の方法

本研究に利用したデータは表 3.1.1 に示す通り，AbemaTV サービスの 2016 年 5 月 1 日から 2016 年 6 月 2 日までの利用者のうち約 110 万人分のデータをランダムサンプリングしたものである．Hadoop[*7]上の Hive テーブルに蓄積された AbemaTV の行動ログから，データ分析として用いることができるように HiveQL で集計・抽出・整形を行いデータを得た．ログ収集・解析システムの概要を図 3.1.1 に示す．

3.2　モデル

ある週のユーザのアクティブ度を p とすると，翌週アクティブ日数 Y は以下のような二項分布に従うと仮定する．

$$Y \sim \mathrm{Binom}(7, p)$$

アクティブ度 p は関数 $f(\boldsymbol{x})$ で得られる実数値を logit 変換することで 0.0 から 1.0 の範囲の確率値となる．

$$\mathrm{logit}(p) = f(x)$$

ここで，$\boldsymbol{x}$ は説明変数を並べた列ベクトルである．

関数 $f(\boldsymbol{x})$ は線形和の場合では一般化線形モデルとなるが，本研究で用いた MARS モデルでは各説明変数の効果量 β_m に対しさらに hinge 関数からなる $h_m(\boldsymbol{x})$ が乗算されたものとなっている．

$$f(\boldsymbol{x}) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(\boldsymbol{x})$$

ここで，$h_m(\cdot)$ は基底関数，M は分割後の説明変数の数（基底関数を有するモデルの個数）である．1 つの変数 x に対して 1 つの境界点 c が設定され $h^+(x) = \max(0, x - c)$ と $h^-(x) = \max(0, c - x)$ の 2 つの基底関数となるが，まず全ての変数に対して機械的に境界点 c が多数設定され次に下記 Generalized Cross Validation（GCV）の値が最小になるように境界点 c と共に変数選択が行われる．

$$\mathrm{GCV}(m) = \frac{\sum_{i=1}^{N}(y_i - f_m(\boldsymbol{x}_i))^2}{(1 - \frac{m + e \cdot (m-1)/2}{N})^2}$$

ここで，N はデータの総数，$f_m(\cdot)$ は m 個の基底関数を有するモデルである．また，e を含む項はペナルティ項であり通常は $e = 2$ か $e = 3$ が使われる．

[*7] http://hadoop.apache.org

表 3.1.1　データの概要

サービス名	AbemaTV
利用データ期間	2016 年 5 月 1 日〜2016 年 6 月 2 日
分析利用ユーザ数	約 110 万人をランダムサンプリング

MARS モデルの fitting は統計分析ソフトウェア R[*8]の earth ライブラリ[*9]を用いて行った [9, 10].

本モデルは，ある日のアクティブ度が翌週のアクティブ日数を決めるというやや強めの仮定を置いている．この仮定を置くことでユーザの継続・離脱につながる要因を示唆できるとともに，ユーザ毎にアクティブ度を算出できるようになる．このアクティブ度はサービスの現状把握などにも利用できるためビジネス現場で使い勝手の良いモデルとなっている．

3.3　利用した変数

MARS モデルで翌週のアクティブ化日数をモデリングする際に利用した変数を表 3.1.2 に示す．翌週アクティブ日数が目的変数であり．それ以外の変数は全て説明変数である．変数の概念図を図 3.1.2 に示す．説明変数は，ある特定の 1 日の中での行動と，その日からみて前週 1 週間のアクティブ日数から成る．構築されるモデルはこれらの説明変数から，明日以降の翌週 1 週間のアクティブ化日数を予測するモデルとなっている．

4　結果

データの記述統計を表 3.1.3 および表 3.1.4 に示す．

総視聴時間に着目すると，平均値と中央値とに大きな乖離が見られることと負の値をとらない量であるにも関わらず標準偏差が平均値の 2 倍弱であることから，正規分布ではない分布形状であることが容易に想像できる．したがって，この説明変数を用いたパラメトリックな線形モデルを使うべきではない．これが，本研究で分布形状によらないノンパラメトリックな MARS を選択した理由の 1 つである．

MARS モデルに fitting した結果を表 3.1.5 に示す．大まかな傾向として，①PC 視聴ではない (=

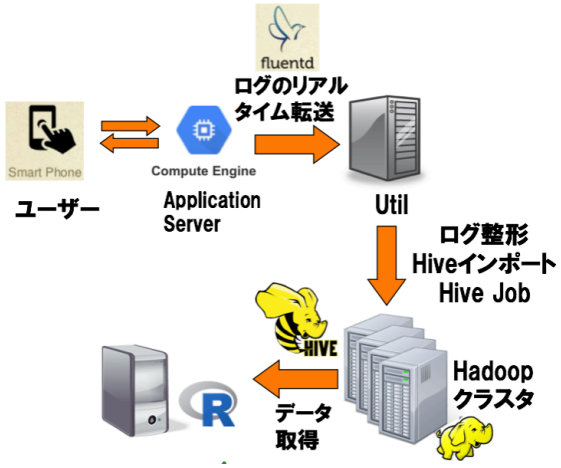


図 3.1.1　ログ収集・解析システム概要

[*8] https://www.R-project.org
[*9] https://CRAN.R-project.org/package=earth

表 3.1.2　利用変数の一覧

変数種別	変数詳細	変数レンジ
目的変数	翌 7 日間アクティブ化日数	0 〜 7(day)
説明変数	前 7 日間アクティブ化日数	0 〜 7(day)
	総視聴時間	0 〜 1437(min)
	視聴チャンネル数	0 〜 20(ch)
	番組予約数	0 〜 508
	プラットフォーム	PC or スマートフォン
	視聴チャンネルカテゴリ	ニュース系 or ドラマ系 or アニメ系 or 音楽系 or スポーツ系 or バラエティ系 or 麻雀系 or その他

表 3.1.3　連続値変数の記述統計

	平均値	中央値	標準偏差	最小値	最大値
翌週アクティブ日数	2.81	2	2.36	0	7
前週アクティブ日数	2.64	2	2.40	0	7
総視聴時間 (分)	43.80	12	88.19	0	1437
視聴チャンネル数	2.06	1	1.63	0	20
予約チャンネル数	0.34	0	2.96	0	508

スマートフォンアプリで視聴) ②アニメ系チャンネルの視聴 ③総視聴時間 35 分以上 ④前週アクティブ日数 1 日以上 ⑤視聴チャンネル数 2 つ以上 ⑥予約番組数 2 つ以上の，① ⑥の特徴をもつユーザのアクティブ度が高い（≒つまり翌週アクティブ日数が増加する）ということがわかる．例えば PC でスポーツ系チャンネルとバラエティ系チャンネルと音楽系チャンネルとを合計 60 分視聴していて前週アクティブ日数が 2 日，予約数が 0 のユーザの場合は表 3.1.5 に記載の効果量を用いた計算によりアクティブ度 0.329（32.9%）であると算出される．

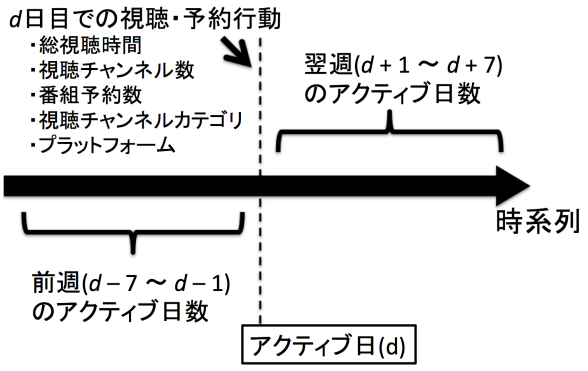


図 3.1.2　利用変数の概要

表 3.1.4 カテゴリカル変数のデータ割合

	TRUE	FALSE
プラットフォーム PC flag	13.5%	86.5%
ニュース系番組視聴 flag	23.7%	76.3%
ドラマ系番組視聴 flag	11.9%	88.1%
麻雀系番組視聴 flag	11.2%	88.8%
アニメ系番組視聴 flag	51.5%	48.5%
バラエティ系番組視聴 flag	21.9%	78.1%
音楽系番組視聴 flag	12.9%	87.1%
スポーツ系番組視聴 flag	19.9%	80.1%
その他番組視聴 flag	18.1%	81.9%

表 3.1.5 MARS モデルによる変数の効果量

変数名	効果量
（切片）	−0.164
プラットフォーム PC flag	−0.406
アニメ系番組視聴 flag	0.172
$h(35 - [\text{総視聴時間}])$	−0.018
$h([\text{総視聴時間}] - 35)$	0.002
$h(1 - [\text{前週アクティブ日数}])$	−0.220
$h([\text{前週アクティブ日数}] - 1)$	0.347
$h(2 - [\text{視聴チャンネル数}])$	−0.225
$h([\text{視聴チャンネル数}] - 2)$	0.055
$h(2 - [\text{予約数}])$	−0.297

5 まとめ

本稿では MARS モデルを用いて AbemaTV サービスにおけるユーザの継続・離脱分析を行いユーザの離脱につながるような行動を示唆することができた．また，同時に個々のユーザのアクティブ度を定量化する方法を提案した．今後もユーザ行動の分析を通してサービスのさらなる発展に貢献していきたい．

参考文献

[1] MMD 研究所：無料動画アプリに関する利用実態調査，`https://mmdlabo.jp/investgation/detail_1453.html` (2015).

[2] 野島豪太，中村陽介，遠藤雅伸，三上浩司，近藤邦雄：アクションポイント制ソーシャルゲームにおける離脱要因の実証実験による検証，2014 年年次大会予稿集，日本デジタルゲーム学会 (2014).

[3] 和田計也, 福田一郎：インターネットテレビにおけるユーザの視聴行動分析，日本マーケティング学会カンファレンス・プロシーディングス，Vol. 5, pp. 62–65 (2016).

[4] Friedman, J. H.: Multivariate adaptive regression splines, *The annals of statistics*, pp. 1–67 (1991).

[5] 片岡弘貴，森田裕之：異常検知を利用した優

良顧客離脱予測モデル, 2011 年秋季全国研究発表大会要旨集, 経営情報学会, p. 78 (2011).

[6] 佐藤哲：計算統計を用いたオンラインゲームユーザのゲーム内行動解析, 情報科学技術フォーラム講演論文集（FIT2013）, Vol. 12, No. 3, pp. 229–230, (2013).

[7] 藤井仰, 森田裕之：年代別購買モデルを用いた CD 購買データの分析, 経営情報学会 全国研究発表大会要旨集 2006 年度秋季全国研究発表大会, p. 8 (2006).

[8] Chou, S., Lee, T., Shao, Y. E. and Chen, I.: Mining the breast cancer pattern using artificial neural networks and multivariate adaptive regression splines, *Expert Systems with Applications*, Vol. 27, No. 1, pp. 133–142 (2004).

[9] Ihaka, R., and R. Gentleman: R: a language for data analysis and graphics, *Journal of Computational and Graphical Statistics*, Vol. 5, No. 3, pp. 299–314, (1996).

[10] Milborrow, S.: earth: Multivariate Adaptive Regression Splines, (2017). R package version 4.6.0, Derived from mda:mars by Trevor Hastie and Rob Tibshirani. Uses Alan Miller's Fortran utilities with Thomas Lumley's leaps wrapper.

3.2 音楽ストリーミングサービス AWA における聴取傾向変化の分析

高野 雅典

概　要　人は音楽聴取行動において，本来の趣味趣向に加え，気分や状況によって聴取する音楽を変える．その聴取傾向は大きな変化から小さな変化までありうる．サブスクリプション型音楽配信サービスにおいてユーザの聴取傾向を定量的に評価することは，ユーザの行動理解や，機能改善・推薦アルゴリズムのユーザ行動に与える影響を知るために重要である．そこで本研究では，ユーザの行動理解を目的とした楽曲聴取行動系列を階層的に分析する手法を提案する．この手法によって，聴取音楽タイプが変わる要因を複数の粒度で把握できる．提案手法は楽曲聴取の系列データについて階層的に隠れマルコフモデルを適用し，聴取行動の大きな変化と小さな変化を検出する．そして検出した楽曲聴取傾向の変化を利用してユーザの行動を分析する．

Keywords　音楽ストリーミングサービス，隠れマルコフモデル，ユーザモデリング

1　はじめに

サブスクリプション型音楽配信アプリケーションは楽曲単位の購入ではなく，1 か月単位の定額契約で大量の楽曲聴取を提供する．したがって従来のビジネスモデルに比べて，ユーザは莫大な量の音楽に容易にアクセスできるようになった．ユーザに膨大な選択肢がある中で快適な楽曲聴取体験を提供するためには，ユーザの聴取行動を知ることが重要である [1].

本研究はこの “聴取傾向の変更” の検出に焦点を当てる．ユーザの楽曲選択には様々な要素が影響する．同じユーザでも本来持つ趣味趣向に加えて様々な外因・内因によって，楽曲の聴取傾向を変える．例えば，いつもはジャズを聴いているが，昨日のテレビ番組の影響で J-POP を聴くなどである．「どのぐらいの頻度でいつ起きるのか？」「アプリケーションのどの機能に起因して起きるのか？」「聴取傾向の変更はよいことなのか？」について知ることが目的である．

例えば，これによって推薦アルゴリズムの発見性・多様性の評価 [2, 3] が期待できる．音楽聴取にクラウドサービスやサブスクリプション型のサービスを利用するユーザはデータをサービスサイドが持っていることから，未知の好みの音楽に出会うことを期待しており [4]，適切なタイミングで適切な楽曲やアーティストを推薦することは重要である．

そこで本研究はユーザの楽曲聴取ログから，そのユーザの聴取傾向とその変化を検出するモデルを提案する [5, 6]．サブスクリプション型音楽配信アプリケーション “AWA[*1]” の聴取ログを利用した．一般に人の行動傾向は様々な粒度で解釈・考察することができる．楽曲聴取行動であれば，1 つの楽曲聴取が最も詳細な粒度であり，最も大きな粒度は邦楽や洋楽などの聴いた楽曲の大きなジャ

[*1] 株式会社 AWA が運営するサブスクリプション型音楽配信サービス（https://awa.fm/）.

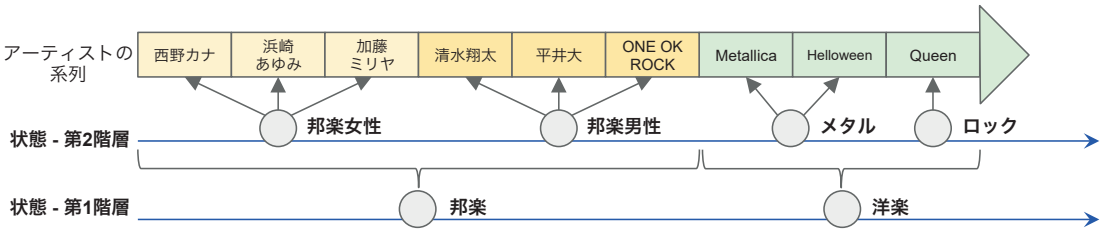


図 3.2.1　モデル概要．第 1 階層はユーザの聴取系列を粗くモデル化し，第 2 階層は第 1 階層の各状態（ここでは邦楽と洋楽）を詳細化したモデル化をする．

ンル分類であろう．したがって人の行動傾向の理解には階層的にモデル化が有効である [7, 8]．

本研究では隠れマルコフモデル（HMM）[9] を使ってユーザの聴取傾向とその変化をモデル化する．複数の粒度を階層的にモデル化するために，HMM を階層的に作成する（図 3.2.1）．構築したモデルを用いて，聴取傾向の変化について分析する．

2　データと手法

本研究では HMM を階層的に用いることによってユーザの聴取傾向を階層的にモデル化した（図 3.2.1）．系列データとして各ユーザ i の各日時 h に最も聴いた[*2]アーティストのリスト $\boldsymbol{x}_i$ を用いた（h は月-日-時間，$\boldsymbol{x}_i = \{x_{ih}\}$）．例えば，あるユーザの系列は “[{5/11-10: 西野カナ }, {5/11-11: 加藤ミリヤ }, {5/11-17: KISS}, {5/13-11: QUEEN}, {5/13-24, AAA}, …]” である（1 つめの項目は 5 月 11 日 10 時台に西野カナ氏の楽曲を最も聴いたことを表す）．音楽を聴取していない時間については系列データに含まない．

最初に大きな粒度として第 1 階層の聴取傾向をモデル化した．学習に利用する系列データは前述の $\boldsymbol{x}_i$ である．HMM の隠れ状態数は 2 から 15 の内，最も AIC [10] が小さいものを採用した．ユーザ i の系列 x_{ih} に対応する第 1 階層の状態を z_{ih} と書く．

次に第 1 階層の結果を用いて，第 2 階層の聴取傾向をモデル化した．学習に利用する系列データはユーザ i の状態 z_{ih} が同じ状態を維持し続けた間を 1 つの系列とした．すなわち，あるユーザの第 1 階層の状態の推移が $\{1, 1, 1, 2, 2, 1, 1\}$ であった場合，状態 1 に対する系列データが 2 つ（$\{1, 1, 1\}$, $\{1, 1\}$），状態 2 が 1 つ（$\{2, 2\}$）が得られることになる．ユーザ i の j 番目の状態 z に対する第 2 階層の状態を z'_{i,z_j} と書く．こちらも第 1 階層と同様に AIC で状態数を決定する．

本手法は再帰的に適用することでアーティストと状態が 1 対 1 に対応するまで階層を深くすることができる．本研究では階層を 2 とした．HMM の学習には Baum–Welch アルゴリズム，推定には Viterbi アルゴリズムを用いた．モデルの作成に使用したデータは月額課金登録ユーザの内，聴取系列長が 70 以上であるユーザからランダムサンプリングした後，その中から典型的なユーザとして系列長が 25%ile から 75%ile の範囲のユーザの系列データを用いた（ユーザ数：2,186）．期間は 2017/03/01 から 2017/05/10 の 71 日間である．

3　結果

3.1　モデルの学習結果

前節の方法でモデルを構築した結果，第 1 階層は 11 状態，第 2 階層は各第 1 階層の状態に対してそれぞれ 4，5，7，7，6，8，8，10，8，8，7 個の状態の HMM が得られた．第 1 階層の各状態の代表的なアーティストを表 3.2.1 に示す．

[*2] 楽曲を最後まで再生した場合，聴いたと定義した．

表 3.2.1 各状態の代表的なアーティスト

状態					
1	Relax α Wave	超特急	Madonna	Coldplay	矢沢永吉
2	Zedd	Ed Sheeran	Avicii	Flo Rida	Rihanna
3	Bruno Mars	Justin Bieber	Ariana Grande	Ed Sheeran	Maroon 5
4	ちゃんみな	CREAM	AK-69	ANARCHY	KREVA
5	BIGBANG	Twice	SHINee	防弾少年団	少女時代
6	Ms.OOJA	globe	Superfly	クリス・ハート	lecca
7	浜崎 あゆみ	加藤 ミリヤ	倖田來未	JUJU	BENI
8	清水 翔太	ケツメイシ	GReeeeN	平井 大	西野 カナ
9	三代目 J Soul Brothers from EXILE TRIBE	西野 カナ	Flower	EXILE	EXILE THE SECOND
10	AAA	Acid Black Cherry	AKB48	Da-iCE	Aimer
11	ONE OK ROCK	乃木坂 46	UVERworld	欅坂 46	[Alexandros]

3.2 基本的な聴取傾向

このモデルを用いユーザの聴取傾向を分析する．分析で使用したデータは，系列長が 2 以上である月額課金登録ユーザからランダムサンプリングをしたユーザの系列データである（対象ユーザ数：19,929）．期間は 2017/04/01 から 2017/05/21 の 51 日間である．

ユーザの聴取傾向変化の典型例を図 3.2.2 に示す．聴取傾向の変化が検出されたことと，それがユーザの振る舞いの違いを表現可能であることがわかる．聴取傾向推移率の頻度分布を図 3.2.3 に示す．状態をほとんど変えないユーザと変えるユーザに 2 分化していたことがわかる．

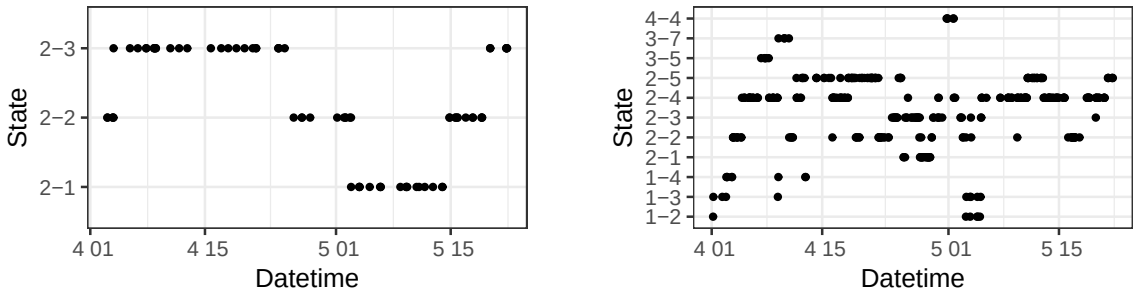
3.3 曜日・時間による聴取傾向への影響

ではユーザはいつ聴取傾向を変化させるのか？それについて知るために図 3.2.4 に各曜日の各時間ごとの状態推移回数の平均値を示す．同図から平日の午前 7–9 時には聴取傾向の変化が発生しやすく，平日の深夜には発生しにくいことがわかる．これは通勤・通学時には多様な楽曲を聴く傾向にあること，または，朝には昨日の聴取傾向とは異なる傾向があることによると考えられる．休日は基本的には平日と同様であるが変化の程度は弱い．これは平日は多くのユーザが通学・通勤しているのに対し，休日はライフスタイルが多様であるため傾向が弱まったと考えられる．

3.4 楽曲選択機能による聴取傾向への影響

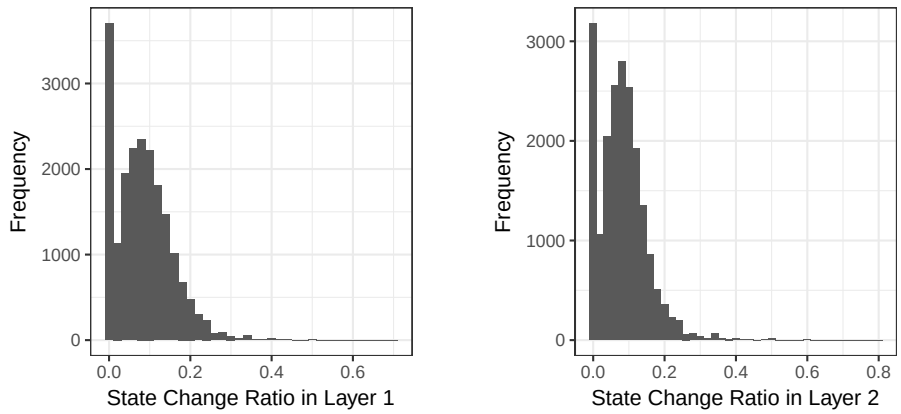
ではどのようにして聴取傾向の変化は発生しているのか？ AWA を含むサブスクリプション型音楽配信サービスでは，複数の楽曲選択機能を提供する．表 3.2.2 に機能ごとの聴取傾向変化数の平均値を示した．各系列の状態にその状態で最もユーザが利用した機能を紐付け，その状態と 1 つ前の状態が異なったときに，その機能によって状態が変化したとみなした．同表から第 1 階層・第 2 階層ともに，オフラインライブラリからの再生（Offline (Track)），自作プレイリスト（My Playlist），ブックマーク（Favorite）によって聴取傾向の変化が発生しやすかったことがわかる．これらはユーザが “自分自身で作成したリストから楽曲やプレイリストなどを選択し聴取する” 機能である．推薦（Recommend）は中間よりやや聴取傾向を変化させやすい傾向にあるといえる．一方で検索（Search）や関連項目[*3]といったユーザ自身が未知の楽曲・アーティスト・プレイリストを探索する行動によっては，聴取傾向の変化は発生しにくかった．また，第 1 階層・第 2 階層の変化しやすさには強い相関があり，どちらか一方のみ

[*3] AWA では楽曲・アーティスト・プレイリストの詳細画面から関連したアーティスト・プレイリストの画面に遷移することができる．



(a) 安定した聴取傾向のユーザ　　(b) 頻繁に変化する聴取傾向のユーザ

図 3.2.2　聴取傾向推移の典型例．縦軸は状態を表す（第 1 階層の状態–第 2 階層の状態）．



(a) 第 1 階層　　(b) 第 2 階層

図 3.2.3　状態変化率（系列長に対する状態推移頻度の比率）の分布．

を変化させやすい機能はなかった．以上からユーザの聴取傾向変化は，ユーザが既知の楽曲・プレイリストの中から能動的に今まで聴いていたものとは異なる傾向のものを選択することに起因していたといえる．また，現状の AWA の機能には聴取傾向の変化の大小どちらかのみが発生しやすいような機能は存在しないといえる．

3.5　聴取傾向変化の利用頻度への影響

聴取傾向の変化はユーザにどのような影響をおよぼすのか？ ここでは「次の 7 日間の楽曲聴取頻度 y」に焦点を当て, 次の一般化線形モデル（GLM）を考える．

$$y \sim B(n, p) \tag{3.2.1}$$
$$\mathrm{logit}(p) = \beta_1 m + \beta_2 s + \beta_3 s' \tag{3.2.2}$$

ここで，$B(n, p)$ は試行数 n，確率 p の二項分布，m は系列長，s は第 1 階層の聴取傾向変化数，s' は第 2 階層の聴取傾向の変化数である．m はサービスの利用頻度を調整するための共変量である．

表 3.2.3 に式 (3.2.1) の分析結果を示す．第 1 階層の変化頻度 s の増加に応じて利用頻度は減少した一方で，第 2 階層の変化頻度 s' の増加に応じて利用頻度も増加したことが分かる．すなわち大きな聴取傾向の変化ではなく，小さな聴取傾向の変化をユーザに提供することがユーザの利用頻度を高めることに重要であることが示唆される．

では多様な楽曲を聴取することはユーザにどのような影響を与えるのか？ 前述のモデルと同様に次の 7 日間の楽曲聴取頻度 y に焦点を当て，次の

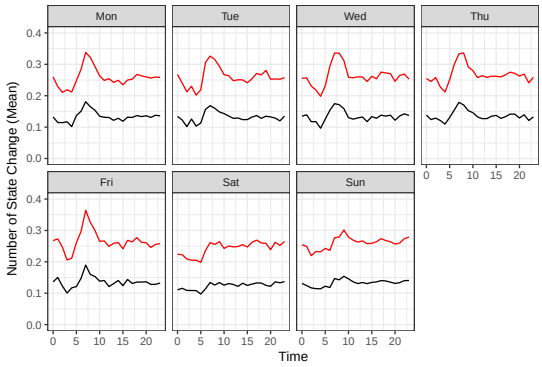


図 3.2.4 各曜日の時間変化（黒: 第 1 階層，赤: 第 2 階層）．集計範囲は 2017/04 である．5 月は祝日を多く含むため除外した．

GLM を考える．

$$y \sim B(n, p) \tag{3.2.3}$$
$$\mathrm{logit}(p) = \beta_1 m + \beta_2 c + \beta_3 c' \tag{3.2.4}$$

ここで c は第 1 階層の期間内に経験した状態の数，c' は第 2 階層の期間内に経験した状態の数である．それ以外は式 (3.2.1) と同様である．

表 3.2.4 に式 (3.2.3) の分析結果を示す．第 1 階層の経験状態数 c の回帰係数 β_2 は有意ではなかった．一方で，第 2 階層の経験状態数 c' の増加に応じて利用頻度も増加したことが分かる．すなわちある程度類似した範囲の楽曲群で様々な楽曲を聴取することがユーザの利用頻度を高めたといえる．

4 まとめ

本研究ではユーザの聴取傾向の変化について分析するために，HMM を階層的に構築するモデルを提案した．その結果，複数の粒度でユーザの聴取傾向の変化を検出できた．

検出した結果を用いて，ユーザ行動を分析した．その結果，ユーザは通学・通勤時に楽曲聴取傾向を変化させがちであること，深夜には楽曲聴取傾向を変化させにくいことがわかった．したがって，平日の午前 7–9 時に普段とは異なった楽曲やアーティストを推薦した場合，ユーザはその楽曲を受け入れやすいことが示唆される．そのためユーザの音楽との出会いを促進するには，この時間帯に施策を実行することが有効であろう．

ただし，聴取傾向変化とユーザの次週の利用頻度の分析によって，楽曲聴取傾向の大きな変化（例えば邦楽女性歌手から洋楽への変化など）はユーザの利用頻度を下げ，小さな変化はユーザの利用頻度を上げることが示された．したがって小さな音楽の出会いを促進し続けることでユーザの満足度を上げ，継続的な利用を促せると考えられる．

これは小さな聴取傾向の変化を発生させやすくする機能の重要性を示唆する．本研究の分析では小さな変化だけを発生させやすい楽曲選択機能は存在しなかった．一般的な楽曲選択機能はブックマークなどの自作のリストや検索などの探索機能である．それらはシンプルで直感的にもわかりやすい機能であるため，“聴取傾向の小さな変化を発生させやすくする”という仕組みを組み込むことは難しい．そのようなユーザの聴取傾向の制御はサービスサイドで提供するほうが容易であろう．例えば聴取傾向の変化を目的とした推薦アルゴリズムの導入などである．

本手法は推薦アルゴリズムの比較・評価にも用いることができる．推薦アルゴリズムにとって「ユーザに新たな情報を提供できるか？」という発見性・多様性の評価は重用である [2, 3]．提案手法を用いることで，推薦アルゴリズムのユーザの聴取傾向をどのように変化させやすいか（ユーザに音楽の出会いを提供できるか）を評価できる．それは推薦アルゴリズムの選択や改善の際に有効な指標になるであろう．

表 3.2.2　再生機能ごとの聴取傾向変化数．全系列データ中に 2000 回以上登場した機能のみ示している．

Function	Layer 1	Layer 2
Offline (Track)	5.17	10.83
My Playlist	4.41	8.88
Favorite (Track)	3.94	8.37
Favorite (Playlist)	2.18	4.09
Recommend	1.70	3.46
Favorite (Album)	1.47	2.66
Favorite (Artist)	1.13	2.10
Genre Rankings	1.04	1.82
Featured Playlists	0.78	1.30
Search (Artist)	0.74	1.31
Artist's Album	0.59	1.06
Search (Playlist)	0.51	0.89
Search (Album)	0.44	0.77
Track Information	0.42	0.78
Search (Track)	0.37	0.65
New Arrivals	0.36	0.66
Artist Information	0.35	0.63
Related Artists	0.31	0.62
Related Playlists	0.19	0.37

表 3.2.3　式 3.2.1 モデルによる分析結果

Explanation Value	Coefficient	Standard Error	t-value	p-value
m	0.0147455	0.0001773	83.157	Less than 2.0×10^{-16}
s	-0.0049439	0.0011958	-4.134	3.56×10^{-5}
s'	0.0045396	0.0014170	3.204	1.36×10^{-3}
Intercept	-1.0551538	0.0111014	-95.047	Less than 2.0×10^{-16}

参考文献

[1] Zhang, B., Kreitz, G., Isaksson, M., Ubillos, J., Urdaneta, G., Pouwelse, J. A. and Epema, D.: Understanding user behavior in Spotify, *2013 Proceedings IEEE INFOCOM*, IEEE, pp. 220–224, doi:10.1109/INFCOM.2013.6566767 (2013).

[2] Ziegler, C., McNee, S. M., Konstan, J. A. and Lausen, G.: Improving recommendation lists through topic diversification, *Proceedings of the 14th international conference on World Wide Web - WWW '05*, New York, New York, USA, ACM Press, p. 22, doi:10.1145/1060745.1060754 (2005).

[3] Hurley, N. J.: Towards diverse recommen-

表 3.2.4 式 3.2.3 モデルによる分析結果

Explanation Value	Coefficient	Standard Error	t-value	p-value
n	0.0141416	0.0001451	97.438	Less than 2.0×10^{-16}
c	0.0052646	0.0067718	0.777	4.36×10^{-1}
c'	0.0071810	0.0025364	2.831	4.64×10^{-3}
Intercept	−1.0903282	0.0146133	−74.612	Less than 2.0×10^{-16}

dation, *RecSys Workshop: Novelty and Diversity in Recommender Systems (Keynote)* (2011).

[4] Lee, J. H., Kim, Y. and Hubbles, C.: A Look at the cloud from both sides now: an analysis of cloud music service usage, *Proceedings of the 17th International Society for Music Information Retrieval Conference*, pp. 299–305, (2016).

[5] 高野 雅典, 鳥海 不二夫, 和田 計也, 福田 一郎: 楽曲聴取行動系列の階層化による聴取傾向変化の検出と行動分析, 第 188 回知能システム研究発表会, (2017).

[6] Takano, M., Mizukami, H., Toriumi, F., Takeuchi, M., Wada, K., Yasuda, M., and Fukuda, I., Analysis of the Changes in Listening Trends of a Music Streaming Service, *2nd International Workshop on Application of Big Data for Computational Social Science (workshop at IEEE BigData2017)* (2017).

[7] Kawazu, H., Toriumi, F., Takano, M., Wada, K. and Fukuda, I.: Analytical method of web user behavior using Hidden Markov Model, *2016 IEEE International Conference on Big Data (Big Data)*, IEEE, pp. 2518–2524, doi:10.1109/BigData.2016.7840891 (2016).

[8] 垣内 弘太, 鳥海 不二夫, 高野 雅典, 和田 計也, 福田 一郎： 潜在状態を用いたコミュニティサービスの分析, 人工知能学会全国大会, 4N2–OS–01b–3in2 (2017).

[9] Rabiner, L.: A tutorial on hidden Markov models and selected applications in speech recognition, *Proceedings of the IEEE*, Vol. 77, No. 2, pp. 257–286 (online), doi:10.1109/5.18626 (1989).

[10] Witowski, V. and Foraita, D. R.: HMMpa: Analysing accelerometer data using hidden Markov models, *R package version 1.0* (2014).

3.3 Web 社会の科学

高野 雅典

概　要　Web を介したオンラインコミュニケーションはいまや私達の日常生活の一部になっており，非常に多くのユーザが存在する．そこでは様々な社会現象が発生している．Web で発生する社会現象の理解・課題解決をするためには，それらに科学的にアプローチしなければならない．我々は Web サービスが生み出すビッグデータを用いてヒトや社会の理解と問題解決に取り組んでいる．本稿では「ヒトの協調メカニズム」，「新たなコミュニケーションツールの登場が社会構造に与える影響」，「仮想社会におけるユーザ間のソーシャルサポート」の 3 つのテーマについて紹介する．

Keywords　計算社会科学，協調行動，社会的グルーミング，ソーシャルサポート，いじめ

1　はじめに

Web で発生している社会問題には，例えば炎上・ネットいじめ・フェイクニュース・社会的分断などが挙げられる．これらは Web の外の社会にも影響を与える．炎上やネットいじめの被害者は精神的・金銭的な被害を被ることもあり [16, 39]，フェイクニュースや社会的分断は選挙の結果にも影響する [18]．これらを解消・緩和し，Web 社会を快適にすることは，Web サービスの活発な利用につながるはずである．したがってこれらを研究し改善策を探ることは Web という世界でビジネスをしている当社にとっても重要である．

ビッグデータを利用した社会科学研究は “計算社会科学” と呼ばれ，近年大きな盛り上がりを見せている [19, 27, 33]．それは Web 社会を理解するだけでなく，これまで観測が難しかったヒトの自由意志に基づく行動の詳細な分析を可能にするため，既存の社会科学を大きく進歩させる可能性がある（図 3.3.1）．また，既存研究の膨大な知見を Web 社会のために利用することもできる．

本稿では秋葉原ラボで行っている Web に関わる社会科学研究について紹介する．まず最初にソーシャルゲームデータを用いた協調行動研究について述べる [36–38]．協調行動は進化生物学，行動経済学，社会心理学などで幅広く研究されている学際的なトピックであり，理論的・実験的な知見が多く存在する [20, 26]．ソーシャルゲーム上での協調行動を分析することで，それらの知見をゲームに適用すること，およびゲーム上での協調行動の知見が既存研究に新たな知見を加えることが期待できる．

次にコミュニケーション方法が社会構造に与える影響の研究について述べる [32, 35]．そこではコミュニケーションの方法がヒトの社会的行動に課す制約とそこから生まれる社会構造をモデル化する．これによって SNS などの新たなコミュニケーションツールの登場がヒトの社会に与える影響について知ることを可能にする．

*1 https://lp.pigg-party.com/

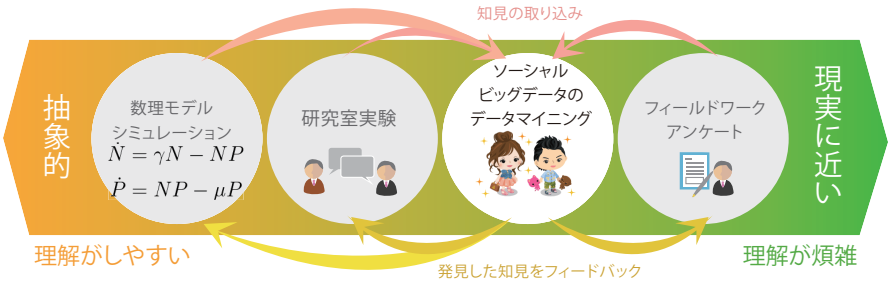


図 3.3.1 ソーシャルビッグデータの分析研究と既存のアプローチ

最後に仮想社会 “ピグパーティ[*1]” のソーシャルサポートの研究について紹介する [34]．ピグパーティには，いじめなど深刻な悩みを相談しているユーザもいる．そのような場があることはユーザの Well-being（幸福度）につながっている可能性がある．この研究では悩み相談がユーザの Well-being に与える影響を検証し，それを促進するためにいつ・どこで・どのようにして起きているのかを分析する．

2 協調行動

相互の協調はヒトをはじめとして多くの動物に見られる現象であり，社会を形成する上で重要な要素である [8, 31]．しかし，他個体に協力する利他的な個体は，利己的な個体と相互作用すると搾取されてしまう．その結果，利己的な個体だけが高い利益を得ることになる．したがって利己的に振る舞うことは最適な戦略であり，相互の協調状態は不安定なはずである [2]．それにもかかわらず我々は社会的生活を営む上で相互に協調し合っている．したがって，ヒトは進化の過程において相互の協調関係を維持するメカニズムを獲得してきたはずである [11]．ヒトやその他の動物の協調行動を説明するために非常に多くの理論的・実験的研究がなされてきた [20, 26]．そのため，安定した相互の協調状態を実現する仕組みや，それを促進するような環境に関する知見が多く存在する．このような協調行動研究は，協調の進化的起源解明という理学的な意義があるとともに，ヒトの協調行動の促進という面で社会的にも重要である．

協調行動の研究では，自分は損をしても相手に利益を与えるという協調行動を定義するために，（囚人のジレンマなどの）個体間の利害が明示的に存在するモデル [29] を作り，その数理・数値解析や心理学実験をすることが多い．したがって，複数のユーザの協調と競争要素が明示的にゲームの仕様に盛り込まれているオンラインゲームは協調行動の研究と相性がよいといえる [36–38]．例えば，協調行動の維持には，協調するか否かといった協調行動に関する戦略とともに，どの相手と相互作用をするかというパートナー選択が重要であることが，多くの理論研究 [3–5, 17, 28]，実験研究 [7, 25, 42] で示されている．

そのような協調行動とパートナー選択行動に関する理論・実験研究で得られた知見の，より現実に近い（モデル化されていない・研究者によって動機づけられていない）環境における成立について証拠を得るために，秋葉原ラボではソーシャルゲーム「ガールフレンド（仮）」のデータを分析した [36–38]．協調行動を維持するためのメカニズムは複数存在する [20, 26]．我々はそのうち環境応答移住 [17, 22, 23] と互恵的利他主義 [40] の 2 つに焦点を当ててそれぞれ分析した．

2.1 環境応答移住

環境応答移住とは，個体が所属するグループ環境に応じてグループを移住するという移住戦略である [17, 22, 23]．利他的な個体は利己的な個体よりも “利己的な個体の増加という環境悪化が与える悪影響” を受けやすい（搾取される）ため

悪い環境から移住をしやすい．その結果，利他的な個体比率が高いグループと利己的な個体比率が高いグループといった偏りが生まれる．利他的な個体は利他的な個体の多いグループに所属するため，利己的な個体からの搾取を避けることができる．ソーシャルゲームではユーザはギルドと呼ばれるグループに所属し，そのグループメンバーと連携してゲームをプレイする．したがってグループに利他的なプレイヤーがいることは重要である．ソーシャルゲームデータの分析の結果，利己的な個体が多いグループに所属する利他的な個体はグループを離脱して環境の良いグループを探していたこと，それによってグループの利他的な個体比率に偏りが生まれたことが示された [36]．すなわちプレイヤーは環境応答移住をして利他的なプレイヤー同士の相互作用（協調）を維持していた．

2.2　互恵的利他主義

互恵的利他主義とは，後で見返りが期待できるならば，即座に自分の利益とならなくても，相手に対して利他的に振る舞うという戦略である．ソーシャルゲームにおいても他プレイヤーに利他的な振る舞いをしたプレイヤーほど，その相手からも利他的に振る舞われていた [38]．すなわちプレイヤーは互恵的利他主義に基づいた協調行動傾向が示された．また，互恵的利他主義を促進させるものとして，相手に直接利益を与える協調行動だけでなく，システム上で明示的な利益もコストも発生しない単純なメッセージ（社会的グルーミング（次節参照））も重要であることが示された．

互恵的利他主義に基づく協調関係を構築するには初対面でも互いに利他的に振る舞う必要がある [2]．しかし初対面の相手に利他的に振る舞うのはリスクが高い．相手が協調的か否かに関する情報がないからである．このような状況でもヒトは初対面において，一切の相互作用なしに利他的に振る舞いがちであることは実験的に示されている [10, 24, 25, 42]．一方で実際にはヒトはまったく相手の情報なしに利他的行動をする意思決定をすることはない．相手を観察し，あいさつや雑談をして協調的な関係を築いていく．前述のようにソーシャルゲームではプレイヤーはグループ内で互恵的な関係を構築しており，また，それがないときには利他的なグループを求めてグループ間移住をする．移住直後には初対面同士での相互作用があるはずである．我々は移住直後にプレイヤーがどのように互恵的な関係を築くいていくかについて分析した [37]．その結果，移住直後のプレイヤーは利他的に振る舞いやすく，また，既存のグループメンバーに対して（互いにコストも利益もない軽量な方法で）コミュニケーションを頻繁に取る傾向があることがわかった．ここからヒトは初対面では軽量なコミュニケーションをすることで相手の協調性に関する不確実性を軽減させていることが示唆される．これは，送信コストの低いシグナルはいくらでも嘘がつけてしまうため競合的な環境では成立しないという理論的な知見 [30] と異なる結果であり興味深い．

まとめ

このように環境応答移住 [36]・互恵的利他主義 [37, 38] ともに，ゲーム内でも有効に働き，ユーザ間の協調行動を促進させていた．

SNS における，情報提供（コメントの投稿）とそれに対する反応（投稿に対する Twitter の “リツイート” や Facebook の “いいね” など）に対して，メタ規範ゲームという枠組みを適用することで，協調行動として扱えるようにした理論研究も存在する [14, 21]．明示的な競争と協調という要素が存在するオンラインゲームだけでなく SNS 上での情報拡散現象にも協調行動研究の知見が活かせる可能性がある．

このようにソーシャルメディアやオンラインゲームのユーザ同士の協力関係について分析することで，ヒトの協調行動についての知見が得られる．また，そこで得られた知見や先行研究の知見をサービスの仕組み作りに活かせるであろう．

3 コミュニケーションツールと社会構造

ヒトの社会構造は，社会関係を構築・維持する社会的グルーミングの進化と発明によって変化してきた．近年の SNS の流行によっても我々の社会は大きく変化したはずである．SNS のような新しいコミュニケーションツールによる社会的グルーミングは，対面の会話や毛づくろいなどの原初的な社会的グルーミングと何が異なり，それによってどのように社会に影響を与えたのか？ 新しい社会的グルーミングの出現が社会構造にどのように影響するかを理解することは，ヒト社会の知識を提供する．

我々は原初的な方法（対面の会話やチャクマヒヒの毛づくろいなど）から現代的な方法（SNS や E-mail）までを含む多様なコミュニケーションデータセットを分析し，それらを説明するモデルを構築した [32, 35]．そのモデルの解析の結果，2 つのタイプの社会的グルーミング（手の込んだ方法と軽量な方法）を発見した（図 3.3.2） [32]．手の込んだ社会的グルーミングは原初的な方法であり，親密な社会関係を構築するのに有効である．一方で，軽量な社会的グルーミングは多くの弱い社会関係を持つことを容易にする近代的な方法である．

各方法において構築・維持される社会関係の数と強さのトレードオフ $C = Nm^a$ [35] によって両者は区別される．ここで，C は社会的グルーミングに掛けるコスト，N は社会関係の数，m は平均的な社会関係の強さ，a が N と m のトレードオフの強さである．手の込んだ社会的グルーミングは $a < 1$，軽量な社会的グルーミングは $a > 1$ になった．

このトレードオフはその社会的グルーミング方法を使うヒトの微視的な振る舞いに制約を与える．その結果，社会的グルーミングの種類によって巨視的な社会構造も影響を受ける．軽量な社会的グルーミングによる社会構造は手の込んだ社会的グルーミングによるものよりも幅広くかつ薄くなる．現代の複雑な社会において，ヒトは多様な社会関係の構築・維持のために両者の方法を効果的に使用していると考えられる．ヒトは相手との社会関係が強い（親密である）ほど利他的に振る舞いがちであるが，1 人が協力できる回数には限りがあるためである [12, 13, 37, 38]．一方で，弱くても多くの社会関係を持つことは情報収集の上で重要である [6, 9]．なぜならば弱い社会関係（弱い紐帯; weak tie）を持つ 2 者には共通の社会関係が少ないため，共通の社会関係の多い強い社会関係に比べて新しく多様な情報を獲得しやすいからである．

4 仮想社会のソーシャルサポート

社会的生活の中でヒトは他者を様々な形で支援し，また支援を受ける．この物質的・心理的支援をソーシャルサポートと言う [41]．ソーシャルサポートのうち相手への共感・愛着・尊敬を示すことによって，相手のストレスが軽減される行為を“情緒的サポート”と言う．

ソーシャルサポートはコンピュータを介したコミュニケーションでも発生する [1, 15]．我々はインターネット上の仮想的な社会“ピグパーティ”（図 3.3.3）におけるソーシャルサポートに焦点を当てて研究した [34]．特にユーザの“現実社会のいじめ”に関する相談に着目した．ピグパーティは若年層に利用者が多く，また，解決すべき非常に重大な問題だからである．

若年層において日本でよく使われている LINE や Facebook は現実の社会関係を多く含み現実の延長線上にある．一方でピグパーティは両者に比べて現実社会とは切り離されており，独自の社会関係が多い．

いじめ相談の効果を分析した結果，相談をしたユーザ・されたユーザともにピグパーティの利用頻度が高くなることがわかった．これはいじめ相談をする・される行動がユーザにとってよい経験であることを示すといえる．

既知の社会関係（友人）が多い閉じたチャットスペースでは，“泣く・生きる・逃げる・嫌・お母さん・お父さん”など感情的な言葉や家族に関す

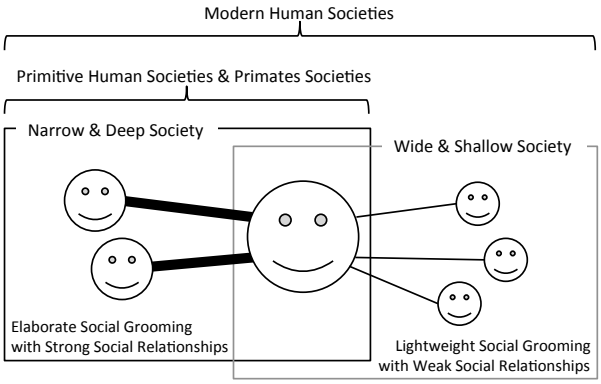


図 3.3.2　コミュニケーション方法と社会構造の関係

図 3.3.3　ピグパーティ

る単語を含む発言が多く見られ，踏み込んだ内容の会話がなされていることが示唆された．一方で比較的開いたチャットスペースでは初対面同士の会話が多く，“いじめ・殴る・学校・先生”などの説明的な単語が多く見られた．

このように仮想的な社会においてもソーシャルサポートが自発的に発生し，それがユーザにとってよい影響を与えていることがわかった．さらに研究を進めこれらを促進する要因が分かれば，それによるユーザの Well-being 向上が期待できる．また，深刻な問題を抱えるユーザに対しては臨床心理士などの専門家による介入というアプローチも考えられる．

5　まとめ

本稿では秋葉原ラボで実施している計算社会科学研究について紹介した．ヒトや Web 社会理解のための基礎研究として協調行動研究と社会構造とコミュニケーションツールの研究を挙げ，ユーザの Well-being 向上を狙った応用研究としてソーシャルサポートの研究を紹介した．

協調行動はソーシャルサポートの一種であるため，協調行動研究は相互にソーシャルサポートし合う関係構築のためのヒトの社会的振る舞いについて知見を与える．また，社会構造とコミュニケーションツールはそのソーシャルサポートに向いているコミュニケーションツールの条件を示唆する．したがって，これらの基礎研究で得られた知見はソーシャルサポートによる Well-being 向上に活かすことができる．また基礎研究による一般性の高い知見は他の Web 社会においても利用可能であろう．

Web はごく最近にヒトが作った社会的空間であり，それがヒトの行動や社会に与える影響は未知なことが多い．またそれを知るためのデータは多くは企業が管理しており，多くはユーザの個人情報を含むため，外部で研究に利用可能なデータは限定的である．したがって，運営企業においても売上やユーザ体験の向上を狙った応用研究だけでなく，ヒトや Web 社会の性質理解を目的とした基礎研究を進め，その知見を論文として公開することが重要であると考えている．その知見によって安全で快適な Web 社会を実現することは，Web 上でビジネスをする企業にとっても良い効果をもたらすはずである．それは基礎・応用研究を発展させうるオープンデータ化にもつながると考えられる．

参考文献

[1] Ando, R., Takahira, M. and Sakamoto, A.: Effects of Internet Use on Junior High School Students' Loneliness and Social Support, *The Japanese Journal of Personality*, Vol. 14, No. 1, pp. 69–79 (2005).

[2] Axelrod, R.: *The Evolution of Cooperation: Revised Edition*, Basic Books (2006).

[3] Chen, X., Fu, F. and Wang, L.: Social Tolerance Allows Cooperation to Prevail in an Adaptive Environment, *Physical Review E*, Vol. 80, No. 5, p. 051104 (2009).

[4] Chen, X., Szolnoki, A. and Perc, M.: Risk-Driven Migration and the Collective-Risk Social Dilemma, *Physical Review E*, Vol. 86, No. 3, p. 036101 (2012).

[5] Damore, J. A. and Gore, J.: A Slowly Evolving Host Moves First in Symbiotic Interactions, *Evolution; International Journal of Organic Evolution*, Vol. 65, No. 8, pp. 2391–8 (2011).

[6] Eagle, N., Macy, M. and Claxton, R.: Network Diversity and Economic Development, *Science*, Vol. 328, No. 5981, pp. 1029–31 (2010).

[7] Fehl, K., van der Post, D. J. and Semmann, D.: Co-Evolution of Behaviour and Social Network Structure Promotes Human Cooperation, *Ecology Letters*, Vol. 14, No. 6, pp. 546–51 (2011).

[8] Fehr, E. and Fischbacher, U.: The Nature of Human Altruism, *Nature*, Vol. 425, No. 23, pp. 785–791 (2003).

[9] Granovetter, M.: The Strength Of Weak Ties, *American Journal of Sociology*, Vol. 78, pp. 1360–1380 (1973).

[10] Grujić, J., Röhl, T., Semmann, D., Milinski, M. and Traulsen, A.: Consistent Strategy Updating in Spatial and Non-Spatial Behavioral Experiments Does Not Promote Cooperation in Social Networks, *PLOS ONE*, Vol. 7, No. 11, p. e47718 (2012).

[11] H, Barkow, J., Cosmides, L. and Tooby, J.: *The Adapted Mind: Evolutionary Psychology and the Generation of Culture*, Oxford University Press (1995).

[12] Haan, M., Kooreman, P. and Riemersma, T.: Friendship in a Public Good Experiment, *IZA Discussion Paper*, Vol. 2108 (2006).

[13] Harrison, F., Sciberras, J. and James, R.: Strength of Social Tie Predicts Cooperative Investment in a Human Social Network., *PLOS ONE*, Vol. 6, No. 3, p. e18338 (2011).

[14] Hirahara, Y., Toriumi, F. and Sugawara, T.: Evolution of Cooperation in SNS-norms Game on Complex Networks and Real Social Networks, *Proceedings of The 6th International Con- ference on Social Informatics (SocInfo)*, Springer International Publishing, pp. 112–120 (2014).

[15] Hobbs, W. R. and Burke, M. K.: Connective Recovery in Social Networks after the Death of a Friend, Vol. 1, p. 92 (2017).

[16] Hosseinmardi, H., Rafiq, R. I., Li, S., Yang, Z., Han, R., Mishra, S., Lv, Q., A Comparison of Common Users across Instagram and Ask.fm to Better Understand Cyberbullying, *2014 IEEE Fourth International Conference on Big Data and Cloud Computing*, pp. 355-362 (2014).

[17] Ichinose, G. and Arita, T.: The Role of Migration and Founder Effect for the Evolution of Cooperation in a Multilevel Selection Context, *Ecological Modelling*, Vol. 210, No. 3, pp. 221–230 (2008).

[18] 小林哲郎, ソーシャルメディアと分断化する社会的リアリティ, 人工知能学会誌, Vol. 27, No. 1, pp. 51–58 (2012).

[19] Lazer, D., Pentland, A., Adamic, L., Aral, S., Barabasi, A.-L., Brewer, D., Christakis, N., Contractor, N., Fowler, J., Gutmann, M., Jebara, T., King, G., Macy, M., Roy, D. and Van Alstyne, M.: Computational Social Science, *Science*, Vol. 323, No. 5915, pp. 721–723 (2009).

[20] Nowak, M. A.: Five Rules for the Evolution of Cooperation, *Science*, Vol. 314, No. 5805, pp. 1560–1563 (2006).

[21] Okada, I., Yamamoto, H., Toriumi, F. and Sasaki, T.: The Effect of Incentives and Meta-incentives on the Evolution of Cooperation, *PLOS Computational Biology*, Vol. 11, No. 5, p. e1004232 (2015).

[22] Pepper, J. W.: Simple Models of Assortment through Environmental Feedback, *Artificial Life*, Vol. 13, No. 1, pp. 1–9 (2007).

[23] Pepper, J. W. and Smuts, B. B.: A Mechanism for the Evolution of Altruism among Nonkin: Positive Assortment through Environmental Feedback, *The American Naturalist*, Vol. 160, No. 2, pp. 205–13 (2002).

[24] Peysakhovich, A. and Rand, D. G.: Habits of Virtue: Creating Norms of Cooperation and Defection in the Laboratory, *SSRN Electronic Journal* (2013).

[25] Rand, D. G., Arbesman, S. and Christakis, N.: Dynamic Social Networks Promote Cooperation in Experiments with Humans, *Proceedings of the National Academy of Sciences*, Vol. 108, No. 48, pp. 19193–19198 (2011).

[26] Rand, D. G. and Nowak, M. A.: Human Cooperation, *Trends in Cognitive Sciences*, Vol. 17, No. 8, pp. 413–425 (2013).

[27] 笹原和俊：私のブックマーク「計算社会科学 (Computational Social Science)」, 人工知能学会誌, Vol. 30, No. 6, pp. 856–859 (2015).

[28] Santos, F., Pacheco, J. and Lenaerts, T.: Cooperation Prevails When Individuals Adjust their Social Ties, *PLOS Computational Biology*, Vol. 2, No. 10, p. e140 (2006).

[29] Smith, J. M.: *Evolution and the Theory of Games*, Cambridge University Press (1982).

[30] Smith, J. M. and Harper, D.: *Animal Signals*, Oxfold University Press (2003).

[31] Smith, J. M. and Szathmáry, E.: *The Origins of Life: From the Birth of Life to the Origin of Language*, Oxford University Press (2000).

[32] Takano, M.: Two Types of Social Grooming discovered in Primitive and Modern Communication Data-Sets (2017). arXiv: 1707.08517

[33] 高野雅典: ソーシャルビッグデータで理解するヒトと社会の性質，電子情報通信学会基礎・境界ソサイエティ Fundamentals Review, Vol. 10, No. 4, pp. 275–281 (2017).

[34] 高野雅典：仮想社会におけるソーシャルサポート効果の検証: ピグパーティにおけるいじめ相談, 教育工学研究会（発表予定）(2017).

[35] Takano, M. and Fukuda, I.: Limitations of Time Resources in Human Relationships Determine Social Structures, *Palgrave Communications*, Vol. 3, p. 17014 (2017).

[36] Takano, M., Wada, K. and Fukuda, I.: Environmentally Driven Migration in a Social Network Game, *Scientific Reports*, Vol. 5, p. 12481 (2015).

[37] Takano, M., Wada, K. and Fukuda, I.: Lightweight Interactions for Reciprocal Cooperation in a Social Network Game, *Proceedings of The 8th International Conference on Social Informatics (SocInfo)*, pp. 125–137 (2016).

[38] Takano, M., Wada, K. and Fukuda, I.: Reciprocal Altruism-based Cooperation in a Social Network Game, *New Generation Computing*, Vol. 34, No. 3, pp. 257–271 (2016).

[39] 田中辰夫, 山口真一: ネット炎上の研究，勁草書房 (2016).

[40] Trivers, R. L.: The Evolution of Reciprocal Altruism, *Quarterly Review of Biology*, Vol. 46, pp. 35–37 (1971).

[41] Turner, R. J., Turner, J. B. and Hale,

W. B.: Social Relationships and Social Support, Springer, Cham, pp. 1–20 (2014).

[42] Wang, J., Suri, S. and Watts, D. J.: Cooperation and Assortativity with Dynamic Partner Updating, *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 109, No. 36, pp. 14363–14368 (2012).

発表一覧

書籍・解説記事

- 高野雅典：ソーシャルビッグデータで理解するヒトと社会の性質，電子情報通信学会基礎・境界ソサイエティ Fundamentals Review， Vol. 10, No. 4, pp. 275–281 (2017).
- R サポーターズ，安部晃生，市川太祐，酒巻隆治，戸嶋龍哉，福島真太朗，和田計也，里洋平，津田真樹：パーフェクト R，技術評論社 (2017).
- 高野雅典：私のブックマーク 「仮想世界の社会科学」，人工知能， Vol. 32, No. 2 (2017).
- 佐藤洋行，原田博植，里洋平，和田計也，早川敦士，倉橋一成，下田倫大，大成浩子，奥野晃裕，中川帝人，長岡裕己，中原誠：改訂 2 版データサイエンティスト養成読本：プロになるためのデータ分析力が身につく！，技術評論社 (2016).
- 福田一郎，鈴木俊裕，善明晃由，高野雅典，藤坂祐介：サイバーエージェントのデータ活用のための R&D 体制と取組み（〈特集〉イノベーションと AI 研究），人工知能， Vol. 30, No. 3, pp. 318–324 (2015).
- 鈴木俊裕，梅田永介，柿島大貴：HBase 徹底入門 Hadoop クラスタによる高速データベースの実現，翔泳社 (2015).
- 高野雅典，和田計也，福田一郎：ソーシャルゲームプレーヤの協調行動の分析（〈特集〉エンターテイメントにおける AI），人工知能， Vol. 30, No. 1, pp. 74–82 (2015).
- 酒巻隆治，里洋平，市川太祐，福島真太朗，安部晃生，和田計也，久本空海，西薗良太：データサイエンティスト養成読本: ビジネスデータ分析の現場で役立つ知識が満載！：R 活用編，技術評論社 (2015).
- 佐藤洋行，原田博植，下田倫大，大成浩子，奥野晃裕，中川帝人，橋本武彦，里洋平，和田計也，早川敦士，倉橋一成：データサイエンティスト養成読本：ビッグデータ時代のビジネスを支えるデータ分析力が身につく！，技術評論社 (2013).
- Sau Sheong Chang(著)，瀬戸山雅人 (翻訳)，河内崇 (翻訳)，高野雅典 (翻訳)，橋本吉治 (翻訳)：R と Ruby によるデータ解析入門，オライリー・ジャパン (2013).

論文誌・国際会議

- M. Takano, H. Mizukami, F. Toriumi, M. Takeuchi, K. Wada, M. Yasuda, and I. Fukuda: "Analysis of the Changes in Listening Trends of a Music Streaming Service," 2nd International Workshop on Application of Big Data for Computational Social Science (workshop at IEEE BigData2017), 2017.
- M. Murase, M. Takano, R. Suzuki, and T. Arita: "A Statistical Analysis of Behavioral Bursts Occurring in a Social Network Game," 2nd International Workshop on Application of Big Data for Computational Social Science (workshop at IEEE BigData2017), 2017.
- M. Takano and I. Fukuda: "Limitations of Time Resources in Human Relationships Determine Social Structures," Palgrave Communications, Vol. 3, 17014, 2017.
- Y. Hirano, F. Toriumi, M. Takano, K. Wada, and I. Fukuda: "Detection of Dangerous Interactions in Online Chat Services," 4th International Workshop —Transformation of human behavior under the influence of Infosocionomics Society—, 2017.
- T. Zenmyo, S. Iijima, and I. Fukuda: "Managing a Complicated Workflow based on Dataflow-based Workflow Scheduler," 2016 IEEE International Conference on Big Data (IEEE BigData 2016), short paper, 2016.
- H. Kawazu, F. Toriumi, M. Takano, K. Wada, and I. Fukuda: "Analytical method of web user behavior using Hidden Markov Model," Workshop Application of Big Data for Computational Social Science (workshop at IEEE BigData2016), 2016.
- M. Takano, K. Wada, and I. Fukuda: "Lightweight Interactions for Reciprocal Cooperation in a Social Network Game," The 8th International Conference on Social Informatics (SocInfo), 2016.
- M. Takano, K. Wada, and I. Fukuda: "Reciprocal Altruism-based Cooperation in a Social Network Game," New Generation Computing, 34, pp. 257–271, 2016.
- R. Y. Shtykh: "Apache Ignite as a Data Processing Hub," In-Memory Computing Summit, 2016.
- 高野雅典, 和田計也, 福田一郎："ソーシャルゲームにおける互恵的利他主義に基づく協調行動", 人工知能学会論文誌, Vol.30, No.6, 2016 (New Generation Computing の連携論文; 和訳版).
- T. Suzuki and H. Kakishima: "HBase @ CyberAgent," HBaseCon 2015, 2015.
- M. Takano, K. Wada, and I. Fukuda: "How Do Newcomers Blend into a Group?: Study on a Social Network Game," 3rd International Workshop on Data Oriented Constructive Mining and Multi-Agent Simulation (DOCMAS) & 7th International Workshop on Emergent Intelligence on Networked Agents (WEIN) (workshop at WI-IAT 2015), 2015.
- R. Y. Shtykh and M. Makita: "Adopting Semantic Similarity for Utterance Candidates Discovery from Human-to-Human Dialogue Corpus," Future and Emerging Trends in Language Technology (FETLT 2015) Workshop, 2015.
- M. Takano, K. Wada, and I. Fukuda: "Environmentally Driven Migration in a Social Network Game," Scientific Reports, 5, 12481, 2015.
- R. Y. Shtykh and T. Suzuki: "Distributed Data Stream Processing with Onix,"IEEE Inter-

national Conference on Big Data and Cloud Computing, Sydney, Australia, 2014.

国内学会／セミナー

- 高野雅典：“ソーシャルビッグデータ・オープンデータによる社会構造変化の発見”, 第 3 回ウェブサイエンス研究会（招待講演）, 2017.
- 高野雅典, 角田孝昭：“仮想社会におけるソーシャルサポート効果の検証 〜 ピグパーティにおけるいじめ相談 〜”, 教育工学研究会, 2017.
- 角田孝昭：“アメーバブログの分析事例: もっと読みたくなるサービスを届けるためのエントリ分析”, 第 10 回 Web とデータベースに関するフォーラム（WebDB Forum 2017）, 2017.
- 高野雅典, 鳥海不二夫, 和田計也, 福田一郎：“楽曲聴取行動系列の階層化による聴取傾向変化の検出と行動分析”, 第 188 回知能システム研究発表会, 2017.
- 水上ひろき, 尾日向洋皓, 横山潤：“スマートフォンのセンサ情報を用いた屋内来訪検知技術”, マルチメディア，分散，協調とモバイルシンポジウム (DICOMO2017), 5C-2, 2017.
- 高野雅典, 福田一郎：“多様な社会関係維持のための社会的グルーミングの多様性”, 第 31 回人工知能学会全国大会, 4N1-OS-01a-2, 2017.
- 垣内弘太, 鳥海不二夫, 高野雅典, 和田計也, 福田一郎：“潜在状態を用いたコミュニティサービスの分析”, 第 31 回人工知能学会全国大会, 4N2-OS-01b-3in2, 2017.
- 原淳史, 馬場惇, 岩崎祐貴, 田中駿：“Entity Linking を用いたユーザのサイト回遊におけるデモグラフィック推定の検討”, 第 31 回人工知能学会全国大会, 3Q1-9in1, 2017.
- 平野雄一, 鳥海不二夫, 高野雅典, 和田計也, 福田一郎：“オンラインチャットサービスにおける未成年者検出”, 第 31 回人工知能学会全国大会, 2J1-3in2, 2017.
- 鷹雄健, 白井英, 安田征弘, 水上ひろき, 鈴木元也, 木村衆平：“サイバーエージェントの Tableau 利用事例”, Tableau Conference On Tour, Tokyo, ブレイクアウトセッション, 2017.
- 善明晃由, 津田均, 田中克季：“可視化のための非構造データの表化手法”, 第 9 回データ工学と情報マネジメントに関するフォーラム (DEIM2017), 2017.
- 高野雅典, 福田一郎：“社会関係の数と親密さのトレードオフが社会構造に与える影響”, 第一回計算社会科学ワークショップ, 2017.
- 高野雅典：“ヒトと社会を理解するための計算社会科学”, 第 23 回社会情報システム学シンポジウム（基調講演）, 2017.
- 數見拓朗, 角田孝昭：“アメーバブログにおけるスプログの特徴と検知手法の検証”, 人工知能学会合同研究会 第 10 回 データ指向構成マイニングとシミュレーション研究会（SIG-DOCMAS2016）, 2016.
- 和田計也, 福田一郎：“インターネットテレビにおけるユーザの視聴行動分析 —継続・離脱分析とユーザアクティブ度の定量化— ”, マーケティングカンファレンス 2016, Vol. 5, pp. 61–65, 2016.
- 力徳正輝：“楽曲生成モデリングのための音符クラスタリング”, 音楽情報科学研究会（IPSJ-MUS）ポスターセッション, 19, 2016.
- 高野雅典：“ソーシャルなビッグデータによるヒトの社会性と社会構造の分析”, 第 57 回 GRL 浜松セミナー, 静岡大学, 2016.
- 和田計也：“AbemaTV でのデータ分析事例”, 第 9 回 Web とデータベースに関するフォーラム

（WebDB Forum 2016）, 2016.

- 高野雅典：“ソーシャルゲームのユーザ行動データを使った協調行動研究: 互恵関係構築におけるライトなコミュニケーションの効果分析”, 社会情報学会 学会大会 若手プレカンファレンス「ゲームとしての社会／社会としてのゲーム」, 2016.
- 高野雅典, 一ノ瀬元喜：“社会関係の強さに基づく社会的グルーミング戦略の適応性”, 第 13 回ネットワーク生態学シンポジウム, 3, 2016.
- M. Murase, M. Takano, R. Suzuki, and T. Arita: “A statistical analysis of play data in a social network game: Effects of communication on cooperative behavior,” 31st International Congress of Psychology (ICP2016), Poster Presentation in Japanese, RC-05-2, 2016.
- 河津裕貴, 鳥海不二夫, 高野雅典, 和田計也, 福田一郎：“潜在状態ネットワークに基づくソーシャルゲームユーザの行動抽出”, 第 30 回人工知能学会全国大会, 4D4-2, 2016.
- 河津裕貴, 鳥海不二夫, 高野雅典, 和田計也, 福田一郎：“隠れマルコフモデルを用いたソーシャルゲームユーザの分類”, 第 35 回ゲーム情報学研究会, 2016.
- 善明晃由, 津田均：“スキーマ定義に基づく SQL ライクな Key-Value ストアクライアント”, 第 8 回データ工学と情報マネジメントに関するフォーラム (DEIM2016), 2016.
- 牧田光晴, シュティフ・ロマン：“対話ログとユーザ属性情報を用いたリカレントニューラルネットワークによる雑談対話生成方式”, 第 2 回 自然言語処理シンポジウム（ポスター発表）, 2015.
- 高野雅典, 和田計也, 福田一郎：“新参者は如何にして新たなグループになじむのか？ : ソーシャルゲームにおける分析事例”, 第 8 回 Web とデータベースに関するフォーラム（WebDB Forum 2015）, 2015.
- 數見拓朗, 内藤遥：“実システムへ適用可能な引用型スプログの検知手法の検証”, 人工知能学会 合同研究会 第 9 回 データ指向構成マイニングとシミュレーション研究会（SIG-DOCMAS2015）, 2015.
- 和田計也, 福田一郎：“音楽聴き放題サービス AWA におけるレコメンド手法の検討（artist2vec の試み）”, 人工知能学会 合同研究会 第 9 回 データ指向構成マイニングとシミュレーション研究会（SIG-DOCMAS2015）, 2015.
- 高野雅典, 福田一郎：“コミュニケーションアプリケーションにおける社会的グルーミングのコスト分配戦略”, 人工知能学会 合同研究会 第 9 回 データ指向構成マイニングとシミュレーション研究会（SIG-DOCMAS2015）, 2015.
- 佐藤栄一：“広告配信やレコメンデーションにおける配信制御・リアルタイム集計での HBase 活用事例”, Cloudera World Tokyo 2015, 2015.
- 牧田光晴, シュティフ・ロマン：“非タスク指向型対話システムにおける対話ペアの意味的類似性を用いた発話候補の抽出”, 言語理解とコミュニケーション (2015-06-NLC-TL), IEICE-NLC2015-10, pp.55–60, 2015.
- 宮西一徳, 高野雅典, 吉田岳彦：“大規模リワード広告システムにおける行動履歴と広告属性を利用したコンバージョン予測モデルの構築”, 第 29 回人工知能学会全国大会, 1H5-4, 2015.
- 和田計也, 増田早紀, 梅林泰孝：“行動クラスタリングと隠れマルコフモデルを用いたユーザの位置予測モデル”, 第 29 回人工知能学会全国大会, 3C4-1, 2015.
- 原淳史, 高野雅典, Roman Shtykh, 川端貴幸：“インターネット広告におけるコンバージョンに近いユーザの抽出方法の検討”, 第 29 回人工知能学会全国大会, 1H5-4, 2015.

- 平藤燎, 牧田光晴：“形態素解析の動的な辞書拡張によるチャットログからの人名表現抽出”, 第 221 回自然言語処理研究会 (SIG-NL), 2015.
- 高野雅典, 和田計也, 福田一郎：“ソーシャルゲームのプレイヤーデータを使った協調行動メカニズムの研究 ～互恵的利他主義と移住の関連～”, 「ネットワークが創発する知能研究会」・「データ指向構成マイニングとシミュレーション研究会」合同秋合宿, 2014.
- 高野雅典, 和田計也, 福田一郎：“ソーシャルゲームにおける互恵的利他主義に基づく協調行動”, 第 8 回ネットワークが創発する知能研究会（JWEIN2014）, 2014.
- 和田計也, 高野雅典, 福田一郎：“スマートフォンゲームにおけるユーザの離脱・継続行動分析”, 第 12 回エンターテイメントコンピューティングシンポジウム（EC2014）, 2014.
- 鈴木俊裕, 梅田永介：“HBase を用いたグラフ DB「Hornet」の設計と運用”, Hadoop Conference Japan 2014, 2014.
- 福田一郎：“研究所・ラボの作り方”, エンジニアの未来サミット 2014～働く場所の見つけ方、作り方～ パネルディスカッション, 2014.
- 數見拓朗：“アメーバブログにおけるスパムブログ検知: 機械学習を用いたスパムフィルタの開発”, 第 7 回 Web とデータベースに関するフォーラム（WebDB Forum 2014）, 2014.
- 善明晃由, 内藤遥：“Ameba における HBase 活用事例”, Cloudera World Tokyo 2014, B-2, 2014.
- 和田計也：“Ameba における大量データ分析、R の活用事例”, データサイエンティストサミット, 2013.
- 高野雅典：“進化ゲーム理論の考え方でソーシャルゲームの “ソーシャル” を分析する”, データサイエンティストサミット, 2013.
- ユハニ・コノリー, 飯島賢志：“Flume を活用した Ameba における大規模ログ収集システム”, Hadoop Conference Japan 2013 Winter, 2013.
- 善明晃由：“Ameba におけるログ解析基盤の変遷”, 第 6 回 Web とデータベースに関するフォーラム 技術報告セッション (WebDB Forum 2013), 2013.
- 和田計也：“Ameba における RHadoop の活用事例”, Cloudera World Tokyo, 2013.
- 善明晃由, 飯島賢志：“Ameba におけるログ解析基盤 Patriot の活用事例”, Cloudera World Tokyo 2013, 2013.
- 高野雅典, 和田計也, 福田一郎：“進化ゲーム理論の枠組みを用いたソーシャルゲームにおけるユーザの利他的行動の分析”, 数理モデル化と問題解決研究会報告, 2013-MPS-95(20), pp. 1–6, 2013.
- 福田一郎：“Ameba サービスにおける Hadoop 活用事例”, OSCA Hadoop セミナー, 2013.
- シュティフ・ロマン, 牧田光晴：“Akka を用いたデータストリーム処理・解析プラットフォーム”, Scala Conference in Japan 2013, 2013.
- 鈴木俊裕：“HBase at Ameba”, Developers Summit 2013, 2013.
- 牧田光晴, シュティフ・ロマン：“大規模データ解析の Ameba サービスへの適用事例”, 第 5 回 Web とデータベースに関するフォーラム（WebDB Forum 2012）, 2012.
- 後藤正樹, 瀬戸山雅人, 高野雅典, 大関正人：“iPad を利用した授業中に使えるデジタル教科書用後付 LMS「Real-time LMS」の開発と実践”, 日本デジタル教科書学会年次大会発表原稿集, Vol. 1, pp. 22–23, 2012.
- 関喜史, 福田一郎, 松尾豊：“コミュニティ構造を利用した Web サービスにおけるユーザ推薦手法の検討”, 第 26 回人工知能学会全国大会（JSAI2012）, 3E1-R-6-3, 2012.

- 福田一郎, 木村衆平：“大規模データ処理基盤とその適用例”, 日本知能情報ファジィ学会 マルチコアとマーケティング研究グループ 第 1 回講演会, 2012.
- 福田一郎：“Ameba サービスにおける Hadoop および関連プロダクトの活用”, 第 4 回 Web とデータベースに関するフォーラム 特別セッション (WebDB Forum 2011), 2011.
- 服部司, 和田計也：“Ameba の統合ログ解析基盤を使った応用・解析事例”, 第 4 回 Web とデータベースに関するフォーラム 技術報告セッション（WebDB Forum 2011）, 2011.
- 福田一郎：“Hive を用いた Ameba サービスのログ解析共通基盤”, Hadoop Conference Japan 2011, 2011.

CyberAgent 秋葉原ラボ 技術報告
Volume.1

2018年2月1日 初版
CyberAgent 秋葉原ラボ 技術報告編集委員会

CyberAgent