公立はこだて未来大学 2019 年度 システム情報科学実習
グループ報告書

Future University-Hakodate 2019 System Information Science Practice

Group Report

プロジェクト名

AI するディープラーニング

Project Name

AI love Deep Learning

グループ名

グループ A

Group Name

Group A

プロジェクト番号/Project No.

16-A

プロジェクトリーダ/Project Leader

川村周也　Shuya Kawamura

グループリーダ/Group Leader

川上達也　Tatsuya Kawakami

グループメンバ/Group Member

川村周也　Shuya Kawamura
川上達也　Tatsuya Kawakami
前田陸　Riku Maeda
田渕知明　Tomoaki Tabuchi
崎野也真人　Yamato Sakino
坂元優也　Yuya Sakamoto
岩成豪　Gou Iwanari
奥村拓馬　Takuma Okumura

指導教員

竹之内高志　寺沢憲吾　香取勇一　佐々木博昭　冨永敦子　片桐恭弘

Advisor

Takashi Takenouchi　Kengo Terasawa　Yuichi Katori　Hiroaki Sasaki　Atsuko Tominaga
Yasuhiro Katagiri

提出日

2020 年 1 月 31 日

Date of Submission

January 31, 2020

概要

昨今, 急速な人工知能の発達が進んでいる. その中で, Deep Learning の進歩は顕著である. この Deep Learning を応用したサービスの 1 つとしてレコメンド機能がある. レコメンド機能とは, ユーザの行動履歴を利用して興味関心に関連する情報を薦める機能である. この機能は協調フィルタリングが代表として挙げられ, 通販サイトや動画視聴サービスなどで用いられている. 現在, その汎用性と需要の高さから, 今後も活用されていくことが想定される.

その一方, 現実における個人間のレコメンドも頻繁に行われるが, 既存のサービスが提供するものと比べ根拠や妥当性に欠けてしまいがちだと感じられる. このような問題は, 円滑なコミュニケーションの妨げになってしまうだろう.

既存サービスのレコメンド機能では, 独立した各ユーザの行動履歴を元に, レコメンドするシステムである. しかし, 現実におけるレコメンドの円滑化を図るには, 互いの行動履歴を考慮してレコメンドを支援する機能が必要になると考えた. このような機能を持つサービスは未だ存在しない.

前述の問題点を解決するために, AI が各ユーザの視聴記録や好みを考慮し, アニメ情報のレコメンド支援をするアプリケーションを考案した.

レコメンドの流れとして, まずレコメンドに必要なデータとして, ユーザのアニメに対する評価データを収集する. 次に Graph Convolutional-Matrix Completion(GC-MC)[1] という機械学習の手法を用いて, 未評価のアニメに対する尤もらしい予測評価を生成する. 各ユーザの評価データはこの予測評価で補完され, 実評価と予測評価により完全に埋まった評価データを作成する. 最後に, この評価データを交換することで自分が高く評価した作品の中から, 相手が高く評価してくれそうな作品を知ることができる. このように, 各ユーザの視聴記録や好みを考慮してレコメンドを支援するアプリケーションがあれば, 円滑なレコメンドを行えると考えられる.

よって本プロジェクトでは, レコメンド支援機能を持つ iOS アプリケーション「AnImediate」を作成し, 実際にサービスとして展開することを目指してアプリケーション開発を行った. 最終的に, アプリケーション自体は完成したが, 現時点ではサービスの展開まで達していない.

（※文責: 岩成豪）

Abstract

In recent years, rapid development of artificial intelligence has been progressing. Especially, the progress of Deep Learning is remarkable. One of the services using Deep Learning is a recommendation function. It is a function that recommends information related to interests using the user's behavior history. This function is exemplified by collaborative filtering. It is used in mail-order sites and video on demand services. Because of its versatility and high demand, it is expected that it will be used in the future. On the other hand, real recommendations are often made between individuals, but they tend to lack grounds and validity compared to those provided by existing services. Such problems will hinder smooth communication.
The recommendation function of the existing service is a function that makes decisions based on the behavior history of each independent user and makes recommendations. However, when considering real recommendations, it is necessary to have a function that supports the recommendation in consideration of each other's action histories.
In order to solve the above-mentioned problems, we devised an application in which AI supports recommendation of anime data in consideration of each user's viewing record and preference.
I will explain recommendation flow. First, the user's evaluation data of anime is collected as the data necessary for the recommendation. Next, we generate a plausible prediction evaluation for the unrated anime with machine learning. We use Graph Convolutional-Matrix Completion (GC-MC)[1] that is one of a machine learning method. The evaluation data of each user is complemented by this prediction evaluation, and the actual evaluation and Create evaluation data that is completely filled out by predictive evaluation. Finally, by exchanging this evaluation data, you can find out which works you highly value from the works you highly valued. Thus, if there is an application that supports the recommendation in consideration of the viewing records and preferences of each user, it is considered that the recommendation can be made smoothly.
Therefore, in this project, we created an iOS application "AnImediate" with a recommendation support function, and developed an application with the aim of actually deploying it as a service.
Eventually, the application itself is completed, but has not yet reached service deployment.

（※文責: 坂元優也）

目次

第 1 章　はじめに　1
1.1　プロジェクトの目的 1
1.2　背景 1
1.3　現状における問題点 1
1.4　グループ目的 2

第 2 章　課題設定までのプロセス　3
2.1　グループ目標 3
2.1.1　前期グループ目標 3
2.1.2　後期のグループ目標 4
2.2　課題設定 4
2.2.1　前期の課題設定 4
2.2.2　後期の課題設定 5
2.3　担当割り当て 6
2.3.1　前期の担当割り当て 6
2.3.2　後期の担当割り当て 6

第 3 章　活動内容　7
3.1　前期の活動内容 7
3.1.1　テーマの検討 7
3.1.2　勉強会 7
3.1.3　先行サービスの調査 7
3.1.4　手法の検討 8
3.1.5　モデルの作成 8
3.1.6　アプリケーション開発 10
3.2　後期の活動内容 11
3.2.1　データ収集 11
3.2.2　手法の再検討 11
3.2.3　アプリケーションのリファクタ 11
3.2.4　後期のモデルの作成 12

第 4 章　アプリの開発の詳細　13
4.1　要件定義 13
4.2　プロトタイプ作成 13
4.3　アーキテクチャ選定 14
4.4　UI デザイン 15
4.5　技術選定 16
4.5.1　ライブラリ 16

4.5.2 API 17
4.6 アプリの実装 17

第 5 章 モデル構築 **19**
5.1 モデルの概要 19
5.2 データセットの作成 20
5.2.1 スクレイピング 20
5.2.2 ジャンル推定 21
5.3 モデルの実装 21
5.3.1 NMF 21
5.3.2 GC-MC 22

第 6 章 発表の評価 **27**
6.1 中間発表 27
6.1.1 ポスター作成と発表準備 27
6.1.2 スライド作成と発表準備 27
6.1.3 中間発表に対する評価シートの内容 28
6.2 期末発表 29
6.2.1 ポスター作成と発表準備 29
6.2.2 スライド作成と発表準備 30
6.2.3 期末発表に対する評価シートの内容 30
6.3 総評 31

第 7 章 個人活動 **33**
7.1 川上達也（グループリーダー, Swift 班） 33
7.1.1 前期 33
7.1.2 後期 33
7.2 川村周也（Swift 班） 34
7.2.1 前期 34
7.2.2 後期 35
7.3 前田陸（Swift 班） 35
7.3.1 前期 35
7.3.2 後期 36
7.4 田渕知明（Python 班） 36
7.4.1 前期 36
7.4.2 後期 37
7.5 崎野也真人 (Python 班) 38
7.5.1 前期 38
7.5.2 後期 38
7.6 坂元優也（Python 班） 39
7.6.1 前期 39
7.6.2 後期 39
7.7 奥村拓馬（Python 班） 40

7.7.1 前期 40
7.7.2 後期 40
7.8 岩成豪（Python 班） 41
7.8.1 前期 41
7.8.2 後期 41

第 8 章 まとめ **43**
8.1 前期のまとめ 43
8.2 後期のまとめ 43

参考文献 **45**

第 1 章　はじめに

1.1　プロジェクトの目的

昨今, 社会の情報化が加速度的に進展し, 人々はその大量の情報を的確に活用できることが求められている. そして, 様々な情報が混在し, 肥大化し続けている中で, 本質的な内容をより効率よく抽出する手段が今も考えられている. そこで注目されている技術として機械学習がある. 機械学習は大量のデータから有用な規則や特徴を抽出できる.

機械学習の手法の中でも Deep Learning(深層学習) がさらに注目を集め, 様々な企業や団体の研究で導入されるようになった. Deep Learning とは, ニューラルネットワークという人間の神経構造を模した機械学習モデルのうち隠れ層と呼ばれる層をより深く重ね合わせたものを指す. 層を重ねることで, データを分析するために必要な要素とその関係を, より多くかつ複雑に蓄積することができるようになった. Deep Learning の精度は用意するデータ量に左右されるため, 限られたデータ量でより精度を上げるための研究開発が進められている. 本プロジェクトでは, この Deep Learning を用いて日常の問題解決に挑むことを全体の目的とする.

（※文責: 前田陸）

1.2　背景

現在, 1950 年代, 1980 年代に続く第 3 次 AI ブームの到来と言われ, 様々な AI を用いた製品やサービスが作られている. 例えば, ロボット掃除機や, 自動運転車, スマートスピーカー, Web サービスのレコメンド機能などがある. これらは既存の製品に AI を組み合わせることで, より便利にしたり, 手間を省いたり, なんらかの問題を解決するものであることが多い. そこで本プロジェクトの目的である, Deep Learning を用いた日常生活における問題解決のために「個人間におけるアニメのレコメンド支援」に焦点を当てた. 個人間のレコメンドでは, 各個人の好みの違いや互いの行動履歴の把握が困難なことなど, 様々な障害が存在する.

（※文責: 崎野也真人）

1.3　現状における問題点

私たちは, 日常会話の中で様々な作品のレコメンドをしあう機会がある. 本プロジェクトでは, アニメのレコメンドについて考える. アニメは, 2000 年ごろから毎年 200 以上の作品が生み出され, 現在では約 10000 作品ものアニメが存在する. また, アニメを鑑賞する人の中で, いままでに見た作品数が 100 作品を超える人も少なくない. それだけ膨大な量の作品情報を元に人間が正確にレコメンドを行うことは困難である. これらの現状から, レコメンドの際に作品が相手の好みに合致しないことや, すでに見た作品をレコメンドしてしまうなどの問題が生じ, レコメンドがスムーズに進まない場合や, 失敗する場合がある.

（※文責: 川上達也）

1.4　グループ目的

1. 3 で挙げた問題を解決するために, アニメのレコメンドを支援するアプリケーションを開発することにした.

そこで, グループの目的として

- アプリケーション上で使用するアニメ情報の収集
- アニメの視聴管理機能, 評価登録機能を持った iOS アプリケーションの開発
- 実際の評価を入力とし, 未評価作品に対する予測評価を出力するモデルの作成

の 3 つを掲げた.

（※文責: 岩成豪）

第 2 章　課題設定までのプロセス

2.1　グループ目標

2.1.1　前期グループ目標

前期の目標を設定するにあたり, 既存の類似するサービスを調査し, 現状のサービスにおける問題点や参考にすべき点の洗い出しを行い, 他サービスとの差別化すべき点や必要な機能, 技術などを明確にした.

iOS アプリケーションを作成するにあたり, プログラミング言語は Python と Swift を採用した. Python とは機械学習や, Deep learning を行うためのライブラリが充実している言語である.Swift とは, iOS アプリケーションを作成することができるプログラミング言語である. その後, Python 班と Swift 班の 2 グループに分け, それぞれのグループごとに目標を設定した.

- Python 班

  アニメの特徴をベクトル化するために, Word2Vec, Doc2Vec, k 近傍法による 3 つの手法に取り組んだ.

  - Word2Vec

  単語を入力情報とし, 分散表現を得ることができる手法である.

  - Doc2Vec

  単語 ID と文書 ID を入力情報とすることで, 単語の並び順などを考慮したベクトル化表現を入手することができる手法である.

  - k 近傍法

  初歩的な分類アルゴリズム. 多次元空間上に特徴ベクトルをもつラベル不明ノードに, ラベル付き最近傍ノード k 個のラベルを用いて多数決でラベル付けする手法.

  上記の 3 つを用いて, アニメをベクトル表現することで具体的に数値化することを目標とした.

- Swift 班

  実際にレコメンドをしあう場合を想定しながら, アプリケーションに必要な機能についての話し合いを行った.

  話し合いの結果,

- モバイル向けデータベースである Realm を用いたユーザの視聴情報の登録, 管理
- 対面での情報交換を行うための Bluetooth 通信機能
- アプリケーション内で使用するアニメ情報を, 既存のサービスである Annict[3] から API を利用して取得

の 3 点を主な機能として実装し, プロトタイプを完成させることを前期の目標とした.

（※文責: 川上達也）

2.1.2 後期のグループ目標

前期が終了した時点で, Swift 班はアプリケーションのプロトタイプを完成させ, Python 班はレコメンド機能実装のための手法検討を行った.

まず, Python 班はレコメンドシステムに有用なモデルの検討を再度行った. その際に, 前期に検討していたモデルでは, 本プロジェクトで行うレコメンド支援に適していないという結論に至った. それを踏まえ, Swift 班と Python 班それぞれで目標の設定を再度行った.

- Python 班

  レコメンドシステムのモデル実装のための使用技術の再検討を行った. その結果, Graph Convolutional-Matrix Completion(GC-MC)[1] という技術を用いて予測評価を行うモデルが有用であるという結論に至った. Matrix Completion(訳: 行列補完) とは, 要素全てが埋まりきっていない行列を入力として, その埋まっていない箇所を尤もらしい値で補完するという技術である. ただし GC-MC[1] は比較的最近の手法であるため, まずはより簡単な NMF(Nonnegative Matrix Factorization) によるレコメンドシステムの作成し, 行列補完を用いたレコメンドの有用性の評価を行うことにした. NMF と並行して, GC-MC(Graph Convolutional-Matrix Completion)[1] の理解を深め, 最終的には GC-MC[1] を用いたレコメンドシステムを完成させることを後期の目標とした.
- Swift 班

  本実装に向けてプロトタイプの見直しを行った. その結果, 大きく 3 つの改善点が見つかった.
- Bluetooth 通信を行っている際に接続が安定しない
- アプリケーションのプログラムの構造が複雑で処理を追いづらく, バグを生みやすい
- アプリケーションの UI が分かりづらく, 使用しにくい機能やボタンがある

  また, 前期にはモデルが完成しておらず, アプリケーションにも組み込めていなかった. そのため, アプリケーションをリファクタリングしてモデルの組み込み, 完成させることを後期の目標とした.

（※文責: 川上達也）

2.2 課題設定

2.2.1 前期の課題設定

はじめに, グループ内で前期の目標について話し合った. 話し合いの中で下記の課題が挙げられた.

- DeepLearning に関する知識・技術の習得.
- 作成するアプリケーションの検討.
- 先行サービスの調査.
- アプリケーションのプロトタイプ作成.
- 手法の検討.
- データ収集・作成または購入.

上記で設定した課題から具体的な活動方針を決定した. まず, Deep Learning の知識の習得を課題として設定した. 学習教材として, 『Python と Keras による Deep Learning』[2] を用いることとした. プロジェクト時間を使って 1 ヶ月間輪読を行い, その後は各自で学習を進めることとした. 作成するアプリケーションの候補として, レコメンド機能があるアニメの視聴管理があがった. そこで, iOS アプリケーションを作成することとし, 先行サービスの調査を行った. その結果, 既存のサービスのレコメンド機能は, 1 人のユーザが他の全ユーザに対してレコメンドする, または, AI が各ユーザに対してレコメンドをするものであることがわかった. そこで我々の課題を, 1 対 1 のユーザ間で行われるレコメンドにおいて, その支援を実際に行う AI を作ることに決定した. 具体的には, 相手にレコメンドするべきアニメを AI が判別し, そのアニメを表示してくれるものである.

サービスとして発表するために, Swift 班 (アプリケーション作成) と Python 班 (AI 作成) に分かれた. Swift 班はアプリケーションのプロトタイプ作成を課題として設定した. 具体的には, 画面定義書の作成, ユーザのデータやアニメを記録するためのデータベース作成があがった. また, AI が, レコメンドするべきアニメを判別するには相手のアニメ視聴情報が必要となるため, Bluetooth を用いた通信について技術調査をすることとした. 前期の中間発表までにアプリケーションのプロトタイプを作成し, 後期から改良するスケジュールを立てた.

Python 班は実際にレコメンド機能を実装するための手法を検討・実装を課題として設定した. まず, メンバーがそれぞれレコメンドに必要な知識とスキルを習得することとした. その後, レコメンドに使えるアルゴリズムを検討し, 実装することとした. また, 手法を検討するにあたって, アニメに関するデータが必要であった. したがって, データの収集・データセットの作成を Python 班の課題として設定した. 前期にデータ収集の目処が立たなかった場合, 夏期休暇の課題とすることとした. また, Deep Learning に関する知識, 技術についても各自で足りないものを夏期休暇で補うこととした.

（※文責: 田渕知明）

2.2.2 後期の課題設定

グループ内で後期の課題について話し合った. その結果, 下記の課題が挙げられた.

- レコメンドに使うモデルの検討.
- iOS アプリケーションの開発.
- 開発タスクの分散.
- 最終発表会の資料作成.
- 最終報告書の作成.

設定した課題から, Python 班と Swift 班の具体的な活動方針を決定した. レコメンドに使うモデルについては, Python 班が開発することとなった. モデルを作成するにあたって, データベースの作成, ジャンル推定を課題として設定した. データベースは Firebase を用いて作成することにした. Firebase とは Google が提供している, プログラミングを必要としないデータベースである. ジャンル推定は, アニメのあらすじから自動推定するアルゴリズムを実装することにした. また, レコメンドシステムの開発経験があるメンバーがいなかったため, 試験的な実装として NMF を使用し, その後 GC-MC[1] を実装するとことした.

iOS アプリケーションの開発については, 前期に引き続き Swift 班が担当した. 前期に実装しきれなかった機能の開発を課題として設定した. 具体的には, 通信の安定化, UI の変更, モデルの組

み込みである. 通信の安定化については, レコメンドするべきアニメを Bluetooth によって交換したが, 通信の途中でアプリケーションが落ちてしまうバグが発生していた. この機能は, アプリケーションのメイン機能なので最も重要な課題とした. モデルの組み込みの経験があるメンバーがいなかった. そのため, アプリケーションに組み込むための技術調査を課題として設定した.

次に, グループ全体の具体的な活動方針を決定した. 前期では, 全体の進捗を管理していなかった. そのため, タスクがうまく分散できていなかったと考えた. この問題を解決するために, 後期では毎日の進捗をグループリーダーに報告することを課題として設定した. 前期の中間発表の資料作りと, 中間報告書の作成がおろそかになっていたことを踏まえて, 最終発表資料, 最終報告書の作成期間について詳細に決めた. 前期では, 中間発表前に 3 週間ほど作成期間を設けた. しかし, 発表資料と報告書を同時進行で作成したため, ともに改善の時間が多く取れなかった. したがって, 後期では最終発表会の 3 週間前から発表資料を作成し, 発表後に最終報告書の作成を行うことにした.

（※文責: 田渕知明）

2.3 担当割り当て

2.3.1 前期の担当割り当て

最初に, アプリケーション開発 (のち Swift 班)・機械学習を用いたデータセットの収集及びモデル構築 (のちの Python 班)・データセットの管理サーバ構築の 3 班へグループメンバーを割り振りを行った. その後, 仕様や方針を見直し, サーバ構築班を Swift 班へ統合してアプリケーションの作成を優先させた. その後, サーバ構築を Swift 班が行うこととした. 進捗を考慮して Swift 班を 3 人, Python 班を 5 人に再度割り振った. Python 班は機械学習の手法を検討することに主軸を据えて活動した.

（※文責: 奥村拓馬）

2.3.2 後期の担当割り当て

後期は Swift 班 2 人, Python 班 6 人で進行した. Swift 班では Bluetooth を用いた登録情報の交換部分の実装や学習済みデータとの通信の確立, またアプリケーション全体の UI の洗練化やデバッグなど, iOS アプリケーション開発に関わる全般の作業を担当した. Python 班では, データセットの収集及び整形と学習を大きな作業ととらえ, 各自がそれらを分担して行った. データの収集に関しては予想される作業量や残りのスケジュールとの兼ね合いをふまえ, 全員で主に夏期休暇中に行った. 今回の学習の根幹となる行列補完方式については, 全員で論文を読むなどして理論を学び, 実装に取り組んだ. 開発終盤で, Python 班で作成した学習結果と Swift 班で作成したアプリケーションとを連携させた, 最終発表に向け, ポスターやデモアプリケーションの製作, また担当教員を通じての発表練習や意見交換などを行った.

（※文責: 奥村拓馬）

第 3 章　活動内容

3.1　前期の活動内容

3.1.1　テーマの検討

まず本プロジェクトで取り組む活動について, Deep Learning を活用してどのようなことがしたいのかを話し合った. その際, ポストイットを用いて 100 種類以上の題材を提案した. その中で, メンバーごとに気になったものを 2 つ決め, それらについての先行研究, 具体的な取り組み方について考えた. その後, メンバーごとにプレゼンテーションを行った. 各メンバーのプレゼンテーション後, 最も通年のテーマとして十分に妥当と考えられる意見を話し合い, 本プロジェクトで行う活動の内容を 1 つに絞った.

（※文責: 川上達也）

3.1.2　勉強会

前期の初期に行われた勉強会では, 参考書 [2] を用いて Deep Learning の仕組み・原理を実際に実装しながら学んだ. 事前に参考書 [2] の章を 1 つずつ各自自習し, プロジェクト学習の時間内に不明な点を確認しながら主要な部分を章ごとに担当を分けて解説などをした.

（※文責: 崎野也真人）

3.1.3　先行サービスの調査

アプリケーション開発を行うにあたって, 先行サービスの調査を行なった. その結果を表 3.1 に示す. Annict[3], あにこれ [4] は, ユーザが作品に評価をつけ, その情報を他のユーザが参考にすることで, どの作品を見るか決める判断材料を確保できるサービスであった. WATCHA[5] はレコメンド作品情報を AI が提示するものであった. これらの調査を元に我々が開発するアプリケーション, 「AnImediate」を考えた.

各サービスの比較を表 3.1 に示す.

表 3.1　各サービスの比較表

サービス名	AI によるレコメンド	視聴管理	他ユーザの視聴情報の確認	レコメンド支援
Annict	×	○	○	×
あにこれ	×	○	○	×
WATCHA	○	○	○	×
AnImediate	○	○	○	○

（※文責: 田渕知明）

3.1.4 手法の検討

先行研究 [6] の調査を基に, 具体的な手法について検討した. レコメンド機能は, それまでのユーザの特徴情報と全体の特徴情報を照らし合わせることが基本となる. そして, この照らし合わせた情報を用いて, そのときユーザが欲しているものを予測するという仕組みである. つまり入力データには, 薦める物の情報と, それに対するユーザの行動履歴が必要となることが明らかになった.

今回は Deep Learning の使用を念頭に, レコメンド情報の特徴を多次元空間上にベクトルで表現することにした. 実際にレコメンドする際, 両特徴情報をベクトル化したのちに, その間の距離を測ることでユーザに薦めるべきコンテンツかどうかを判定できる仕組みになっている. 具体的には, 重みを一様とした際, ベクトル間の直線距離が短いほど特徴は似通っているといえる. このベクトル化に際して先行研究を参照し, 『Python と Keras による Deep Learning』を利用する. 具体的には, 先にアニメの特徴を多次元空間にベクトル化し, そのベクトルをユーザに当てはめて, ユーザの特徴ベクトルを決めるという手法をとることにした.

さらに本プロジェクトはレコメンドするコンテンツをアニメに絞ることで, 開発の方向性を明確にした. このため, 出演者や制作会社などアニメの特徴を表す情報を増やすことで精度が向上するのではないかと考えられた. これらを踏まえて, 多角的な視点でレコメンド可能なシステムを実装することをプロジェクトの目標とした.

（※文責: 坂元優也）

3.1.5 モデルの作成

レコメンド機能を実現するために, アニメの特徴ベクトルとユーザの特徴ベクトルを同じベクトル空間上に生成することが必要である.

さらにベクトル化のために

- アニメのデータとその視聴者であるユーザがつけた評価のデータを収集
- アニメの内容をデータに活用するために行うあらすじのベクトル化
- データを基にベクトル化するための機械学習モデルの構築

の 3 つが必要になることを手法の検討により明確にした. このため作業を分担し,

- Word2Vec によるアニメのレビュー解析及びそのためのデータ収集
- Doc2Vec によるアニメのあらすじ解析
- k 近傍法によるユーザやアニメの分類表現

の 3 種類の手法の構築を試みた.

（※文責: 奥村拓馬）

データの収集

これまで, 各アニメのレビューを収集し, Word2Vec により文章を単語に分割, 単語の出現頻度解析, 単語の類似度推定を目的として活動を行った. また ウェブスクレイピングによる Web 上のデータを収集した. 前期時点でのデータセット生成プログラムの仕様は次の通りであった.

1. ブラウザを起動し, 目的となるデータ（レビュー）が掲載されたページへアクセスする.
2. レビューを全て表示させる.
3. レビューを抽出する.
4. 抽出した結果をファイルに保存する.
5. データ整形を行う.

用いている手法では, 学習に無関係な文字列 (html のタグやレビュー点数など) までファイルに出力されるため, 出力されたファイルから不要な文字列を消去するなどの整形を行った. これを怠ると, 無関係な文字を学習し, 出力に影響を及ぼすおそれがあるためである.

現実には, アニメごとのレビュー数, 文章量に偏りが生じるため, 充分なデータが準備できないアニメも多数存在することが推定された. よって, 出力について考察し修正するための期間を設ける必要があると考えられた.

（※文責: 奥村拓馬）

あらすじのベクトル化

アニメのあらすじをベクトル化するために Doc2Vec を利用した. これは文章の分散表現を得る手法である. 他に分散表現を得る手法として Word2Vec がある. 本節では, Word2Vec と Doc2Vec の違いについて明らかにし, あらすじのベクトル化における Doc2Vec の有用性について述べる.

Word2Vec は単語の分散表現を得る手法である. Word2Vec のモデルには CBoW モデルと Skip-gram モデルの 2 つがある. CBoW モデルは周辺の単語から対象の単語を推測する. Skip-gram モデルは対象の単語から周辺の単語を推測していく. 両手法共に入力は単語を one-hot 表現した単語 ID である. これらのモデルを用いることで単語をベクトル化でき, 単語間類似度などの計算ができるようになると考えられた.

一方で Doc2Vec が Word2Vec と異なる点は入力にある. Word2Vec では入力は単語 ID のみであった. しかし, Doc2Vec ではこれに加えて文書 ID を入力とすることにより, 単語の並び順などを考慮しベクトル化できる. このため, 異なる文書同士の類似度が計算できるようになった. よって, Doc2Vec を用いてあらすじのベクトル化し, 文章間類似度を計算することにより, 類似したジャンルのアニメ間で高い値が出力されるようになった.

（※文責: 岩成豪）

k 近傍法による分類とベクトル化

レコメンドアルゴリズムは, データから得た情報を解析し, その要素間の関連度を出力する必要がある. そのため, アニメをそれぞれ作品ごとにベクトル化するよう, まずは分類アルゴリズムの基本となる k 近傍法を用いた. 学習用のデータには, ユーザのアニメに対する評価をアニメごとに表した 2 次元配列を利用した.

k 近傍法とは, 要素 A の分類時に, A に最もベクトルの直線距離が近いデータを k 個選び, その k 個のデータから多数決で分類するアルゴリズムである. これは分類アルゴリズムの中でも極めて単純なアルゴリズムで, 計算量が多いため怠惰学習とも呼ばれるものである. 前期ではデータ整形後のテスト用に, Python で使用可能な scikit-learn ライブラリの sklearn. neighbors モジュール内にある NearestNeighbors[7] を使用した. この関数は 1 つのラベルが未確定なノードを, 最も近い k 個のノードの加重平均やユークリッド距離を利用し, ラベルとなる数値 (ベクトル) としてを算出するものである. それぞれの成分にラベルの代わりとなる変数があらかじめ埋め込まれており, 最近傍かつラベルが明確なノードを選んで分類を行う. 図 3.1 の例では, 三角ノードと四角ノードはそれぞれのクラスを表しており, 重みは一様と仮定する. このとき中心の十字のノードは $k = 3$ のとき四角ノード一つ, 三角ノード二つのため, 三角ノードに分類される. $k = 6$ のときに四角ノード四つ, 三角ノード二つのため, 四角ノードに分類される.

これにより, アニメのベクトル化を実現できる. これにより, あらすじ解析や知識的な要素では予測できないアニメの特徴を捉えることが可能になると考えられる.

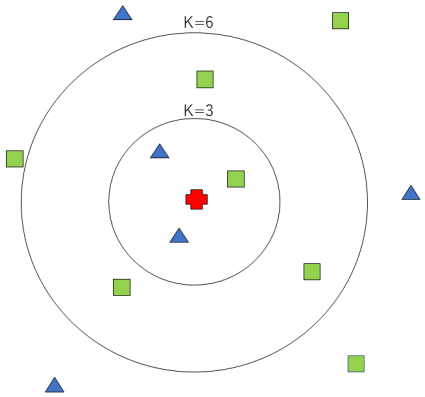

図 3.1　k 近傍法の視覚的表現

（※文責: 坂元優也）

3.1.6　アプリケーション開発

アニメのレコメンド支援というテーマをもとに, アプリケーションの概要を決定した. 具体的には, アニメの視聴管理機能をベースに, Bluetooth を用いてユーザ間で視聴情報を交換するといった基本的な機能を決定した. また, ただ AI がレコメンドを提案するのではなく, それぞれのユーザが他者へ勧めるべき情報を提案するという点で他のレコメンドサービスとの差別化を図った.

基本的な機能を決定した後, 画面定義書を作成し, 各画面ごとに詳細な機能を決定した. 画面定

義書の作成にあたって, Illustrator を用いて実際のアプリケーション画面に近い, デザインプロトタイプを作成した. 開発の前に技術選定を行い, ログイン機能に Firebase, データ管理にモバイルデータベースである Realm, その他画面ごとに必要なライブラリなどを調べ, 決定した. また, アプリケーション上で必要なアニメの情報はアニメ視聴管理サービスである Annict[3] が提供している API から取得した.

開発は, 1 画面ごとに担当者を決め, GitHub を用いてソース管理を行うことで分担しながら作業を進めた. 機能単位で開発を行い, 完成したら全体へマージしてテスト, デバッグというようにアジャイル手法で開発を進めた. また, 実機でテストをしながら使いづらいと感じた部分や, 変更が必要だと感じた部分は即座に他の班員と共有し, 話し合いながら臨機応変に対応した.

（※文責: 川村周也）

3.2 後期の活動内容

3.2.1 データ収集

本プロジェクトでは, モデルの学習に必要となる訓練データとモデルの精度向上のために必要となる補助情報の 2 つを集めた. 訓練データは, あにこれ [4](Web サイト) からアニメの評価データをスクレイピングし, 収集, 記録した. 補助情報は, Firebase(データベース) を利用し, Firebase 内にアニメの各情報 (制作会社名, 監督名, 放送年度, タイトル名, あらすじ) を公式サイトや Wikipedia などから, 収集し, 記録を行った.

（※文責: 岩成豪）

3.2.2 手法の再検討

前期では, アニメの特徴ベクトルとユーザの特徴ベクトルを同じベクトル空間上に配置することを目指していた. 具体的には, ユーザがアニメを評価した評価データの収集, アニメの特徴ベクトルに活用するためのあらすじなどの情報, これらのデータを元にベクトル化するためのモデルの構築を必要とした. これにより, Word2Vec や k 近傍法などの手法を用いることを想定していた.

しかし, データ収集やベクトル化の成功に見通しが立たない状態が続き, 後期で手法の再検討を行った. 再検討の結果, 行列補完手法の 1 つである GC-MC[1] がより適しているとの結論に至った. 行列補完を用いた予測評価生成がアプリケーションの趣旨と相性が良く, なおかつ GC-MC[1] は補助情報による精度の向上が可能なモデルであることが理由である. これらにより, GC-MC[1] が採用されることになった. また, 当初この手法は実現が難しいと判断していたため, NMF での試験的実装も想定した.

（※文責: 岩成豪）

3.2.3 アプリケーションのリファクタ

アーキテクチャとして Redux(4 章参照) を採用し, 状態の管理をしやすくした.

今回のアプリケーションでは,

- 相手のアカウント情報を受け取った.
- 相手の視聴情報を受け取った.
- 自分のアカウント情報を送信した.
- 自分の視聴情報を送信した.

の 4 つの状態を満たした時に通信を終了して次の画面に遷移するという処理を行っていた. しかし, 通信が非同期であるため状態の管理が上手くいっておらず, データがないのに画面遷移してアプリケーションがクラッシュしたり, 通信画面からずっと遷移しないなどの問題が発生した. そこで, Redux と Rx(4 章参照) というライブラリを導入し, 状態の変化を単方向のデータフローとして管理することでそれぞれのクラスの責務が明確になった. これにより複雑な処理の可読性をあげたり, 通信を安定して行うことができるようになった.

（※文責: 崎野也真人）

3.2.4 後期のモデルの作成

モデルをアプリケーション用に編集し, 組み込むことがモデル作成の最終目標であった.

まず初めに GC-MC[1] の論文の輪読を試みたが, 内容が難解なためにこれは現実的ではなかった. そのため役割分担したうえで, その利用方法を探ることになった. GC-MC[1] を使用するにあたって利点となったことは信頼のおける既存のプログラムがあったことだった. よってこのプログラムと論文を併用しモデル作成を行うことにした. 具体的な手順としては論文を読んだ後に, プログラムをアプリケーションで使うためのデータセット用に変更を加えた. この改変したプログラムをアプリケーションに組み込めるように改良し, その後, 精度についても検討を行った. モデルの詳細は 5 章で述べる.

（※文責: 坂元優也）

第 4 章　アプリの開発の詳細

4.1　要件定義

実際にアプリケーションの開発を始める前に要件定義を行い, 仕様を決定した. その際に決定した項目は以下のとおりである.

- 機能
  個人間のレコメンド支援をアプリケーションとして実装する上で,
  – アニメの視聴管理機能.
  – 各アニメに対するユーザの評価機能.
  – ユーザ間で情報を交換する通信機能.
  – ユーザがデータベース内を検索するための機能.
  の 4 つの機能が必要だと考えた.
  まず, ユーザは各アニメについて, 「見た」「見たい」「見てる」「見てた」の 4 つのステータスから 1 つを登録する. 「見た」に登録したアニメについては 5 段階で評価し, これを入力としてアプリケーションに組み込まれたモデルで予測評価を出力する. 視聴情報と評価情報を交換し, 各ユーザの情報から, 自分が高評価をつけた作品の中で相手が未視聴かつ高評価をつけそうな作品をレコメンドする. また, ユーザによる入力作業を伴うことから利便性を向上させるため, データベース上のアニメをジャンル, タイトル, 放送年などで検索できる機能を実装した.
  決定した機能を元に技術選定も行った. アニメ情報は全ユーザがアクセスするためのデータベースである Firebase に登録し, 各ユーザ情報については, モバイル向けデータベースである Realm を用いてローカル端末内に保存することにした. 交換機能は対面を想定し, 将来的にすれ違いの通信を実装することも視野に入れて Bluetooth 通信を採用した.
- UI
  機能を元に UI・UX デザインを行い, 画面定義書を作成した.
- 必要なデータ
  アプリケーションに表示させるための各アニメ情報, 学習のための評価データ.

（※文責: 奥村拓馬）

4.2　プロトタイプ作成

中間発表の際に, アプリケーションのデモンストレーションを行うことになっていたため, 早期からプロトタイプの開発に取り掛かった. プロトタイプ開発の段階では, 使用するアーキテクチャや実装方法にルールを設けておらず, 開発速度のみを重視した開発となった. アプリケーション内部の主要な機能は, 要件定義の内に確定させていた. プロトタイプの段階で実装した主要機能は, Firebase によるログイン機能, Realm によるアニメの視聴情報管理機能, Bluetooth によるユーザー間の視聴情報交換機能であった. その他に, アニメのランキング表示, アニメをタイトルや年代

による検索ができるようにした. Firebase と Annict[3] の API によるアニメのデータベース作成もプロトタイプの段階で完了した. 上記の機能を約 1 ヶ月で開発することとなった為, 各画面 UI の不足や各種エラー, Bluetooth による通信不良などが発生し, アプリケーションの状態は万全とはいかなかった. 中間発表のデモンストレーションではアプリケーションがクラッシュすることは無く, プロトタイプ開発は完了した.

プロトタイプ開発の際に, アーキテクチャや実装方法にルールを設けることは無かったが, グループ開発としては最適な手順で進めることができた. 各画面, 各機能ごとに担当者を予め決定し, Github を用いてソース管理を徹底しながら開発した. 新規の機能を実装するごとに, 変更箇所をマージ, テスト, デバッグを繰り返し, 致命的なエラーを発生させないように心掛けた.

（※文責: 崎野也真人）

4.3 アーキテクチャ選定

前期はアーキテクチャに関しての知識が足りなかったため, 基本的には MVC を採用し, 開発を進めた. 今回作成したアプリケーションはユーザ間での通信や Firebase との通信を頻繁に行うため, 状態管理のためのフラグを各画面ごとに作成, 管理していた. そのため, バグを生みやすく, 通信が安定しないなどの問題が生じた. 後期はこの問題を解決するために, Redux アーキテクチャへと移行した. Redux は UI 構築のための JavaScript ライブラリである. React において UI の State(状態) を管理するためのフレームワークであり, それを Swift で実現するための ReSwift というライブラリを用いて実装した.

Redux は以下の 3 つの原則を遵守するように設計されている.

- 信頼できる唯一の情報源.
- State は読み取り専用.
- 変更はすべて純粋関数で行われる.

1 つ目に, アプリケーション全体の状態を 1 つのオブジェクトツリーで表現する. 2 つ目に, 状態はイミュータブル (不変) であり, 既存の状態を変更するのではなく, 新しく状態を作成する. 3 つ目に, 変更は全て純粋関数によって行われる. 参照するインスタンスとしての状態は不変だが, 新しい状態を参照することで, アプリケーションとしては非破壊, 変更不可を実現する.

上記の原則を踏まえて, データフローを単方向に限定する (図 4.1) ことにより, 処理を追いやすくなり, 作用を把握しやすくなるという利点がある.

Redux の各機能についての詳細な説明は以下の通りである.

- Store

  アプリケーション内に 1 つのみ存在し, State を保持する. State へアクセスする get State 関数と State を変更するための dispatch 関数を提供する.

- State

  各画面につき 1 つの State をもち, 構造体の木構造で表現される.

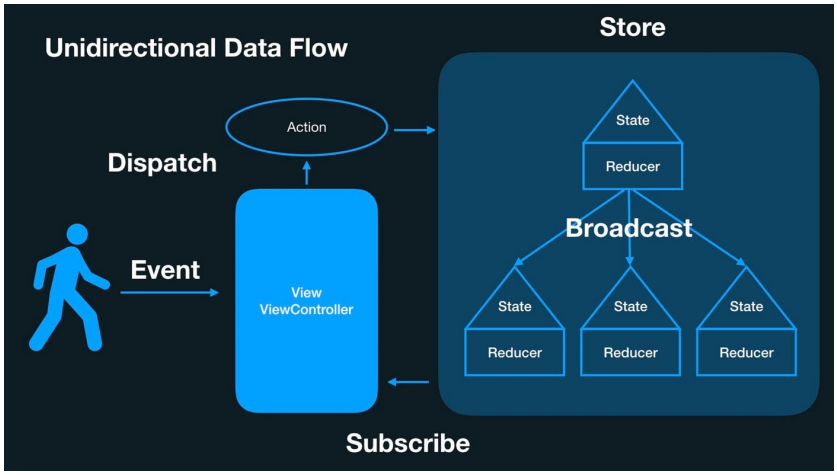

図 4.1　Redux におけるデータフロー [8]

- Action

  State を変更するための指示を表現する, 処理を有しない単なるデータである. Reducer で行う処理の種類であり, そのインプットデータとなる.

- ActionCreator

  Redux では Reducer は純粋関数であり, 副作用を起こす処理を行うことはアンチパターンである. そのため, 非同期通信など, 副作用を伴う処理はここで行う. 副作用が無いとは, 同じ条件下で常に同じ結果を返すこと, また, 他のいかなる機能の結果にも影響を及ぼさないことをいう.

- Reducer

  Action と現在の State を引数にとり, 変更させた新しい状態を返す純粋関数.

- ViewController

  Store の dispatch 関数を用いて Action を発行する. 変更された State は RxSwift によって監視され, 常に最新の状態が反映される.

（※文責: 川村周也）

4.4　UI デザイン

後期には, アプリケーションのリファクタリングに伴い, UI の全面的な見直しを行った. データ収集によって表示できる情報が増えたこと, ユーザのアイコンとアニメのサムネイルの UI など使い回しが多かったことから, 改めて各画面ごとにデザインし直した.

（※文責: 川村周也）

4.5 技術選定

4.5.1 ライブラリ

iOS アプリケーション開発で使用したライブラリは以下の通りである.

- ReSwift

  Redux を Swift で実現するためのライブラリである.

- RxSwift, RxCocoa, RxDataSources

  Rx とは Reactive Extension の略で, RxSwift はその Swift 実装である.

  Observer パターンというデザインパターンが Rx における処理の基本的な設計として用いられている. デザインパターンとは, オブジェクト指向プログラミングの中で再利用可能な設計手法のことである.

  Observer パターンは, 観察される側 (Subject) と観察する側 (Obsever) の 2 つの役割が存在し, Subject の状態が変化した際に Observer に通知され, 状態変化に応じた処理を実装することができる. このデータの流れは Stream と呼ばれ, 時系列順に並んだ進行中のイベントを表す. これによって, 非同期処理や時間に関するイベント処理など煩雑になりがちな実装を簡潔に記述できる.

  RxCocoa は UIKit を扱うために必要で, RxDataSources は TableView や CollectionView などのリスト表示させるデータを扱い, データが変更されたタイミングで画面を更新し, 変更内容を反映させることができる.

- Swinject, SwinjectStoryBoard

  DI(Dependency injection) と呼ばれる概念を実現するための DI コンテナという役割を果たすライブラリ. DI は依存オブジェクトの注入と訳され, オブジェクトの生成と使用を分離することで, 2 つのオブジェクトの依存関係を疎にできる. ここで言う依存とは, オブジェクト A が内部でオブジェクト B を使用しているとき, A が B に依存しているということである.

  今回のアプリケーションでは Redux の ActionCreator を DI で実装している. ActionCreator では外部と非同期通信を行っているため, ここに DI を採用することで通信先をモックと本番で切り替えることなどが容易になる. 複数個のオブジェクトを 1 つのコンポーネントで管理することで生成されたオブジェクトを保持しておくことができ, 同じオブジェクトを再度生成するというコストを削減することもできる.

- SwiftGen

  リソースに関わるコードを自動生成してくれるライブラリ. 画像, フォント, カラーなど文字列で指定するようなリソースから自動で Swift ファイルを生成してくれる. 各ファイルに散らばるハードコーディングをなくし, 運用保守がしやすくなる, タイプミスをなくせる, などとったメリットがある.

- Sourcery

  Stencil という Swift のためのテンプレート言語を用いて作成されたテンプレートの通り

にコードを自動生成してくれるライブラリ．後期にリファクタリングするにあたり, Redux アーキテクチャを採用した．Redux では各画面ごとに State や Action, Reducer などを作成する必要がある．画面ごとに使う変数名やクラス名は異なるが, 処理自体はデザインパターンに則ったものであるため, ほとんど同じである．リファクタリングの効率化とヒューマンエラーの軽減を目的としてこのライブラリを導入した．これによって作業時間が大幅に削減できた．また, 各変数の名前などを書き換えるだけで良いので余計なエラーが発生することもなかった．

- Firebase

  Google の提供している, mBaaS(mobile backend as a Service) と呼ばれるサービスの 1 種である．mBaaS は, サーバサイドのプログラミングを必要とせず, クライアントサイドで提供されているライブラリから関数を呼ぶことで簡単にバックエンドの機能をアプリケーションに組み込むことできる．また, Firebase 内にもいくつかのサービスがあり, 今回使用したのは, RealtimeDatabase, FireStorage, MLKit である．RealtimeDatabase は NoSQL のクラウドデータベースで, データの保管と同期を行うことができる．データはすべてのクライアントに渡ってリアルタイムで同期され, アプリケーションがオフラインになっても利用できる．

（※文責: 川村周也）

4.5.2 API

API(Application Programming Interface) とは, あるコンピュータプログラムの機能や管理するデータなどを, 外部の他のプログラムから呼び出して利用するインターフェースの仕様である．今回のアプリケーションでは, アニメ情報を収集する際に, Annict[3] というアニメの視聴管理サービスの API を使用した．Annict[3] の API ではアニメの名称, 放送年, 制作会社, キャスト, スタッフ, 各話の情報などが提供されていた．

（※文責: 前田陸）

4.6 アプリの実装

前期は基本的に MVC のアーキテクチャを意識しながらプロトタイプ開発を進めた．

まず初めに, Annict[3] の提供する API, スクレイピング で収集したデータを元に Firebase 上にアニメのデータベースを作成した．そして Firebase からデータを取得し, Realm で「見た」「見ている」「見たい」「見ていた」の 4 つの視聴ステータスに分類できるような視聴管理アプリケーションとしての機能を作成した．

UI の作成は画面定義書を元に組み立てることでスムーズに作業を進めることができた．次に検索機能の実装を行った．タイトルや放送年で検索できるようにすることで視聴管理機能の利便性が向上した．そして交換機能を実装した．

前期の時点ではレコメンドモデルができていなかったので視聴情報を交換し, 互いに片方しか見ていないアニメと両方が見たアニメを結果として表示させた．

後期は, 通信が安定せず, アプリケーションが頻繁にクラッシュするといった問題を解決するた

めにアーキテクチャを見直し, Redux を採用した. また, 大量のファイルを新たに生成する必要があったため, メタプログラミングを導入し, ファイルとコードを自動生成することで作業の効率化を図った. Python 班が作成したモデルを組み込む作業も行った. モデルは TFLite 形式のモデルを Firebase の機能の 1 つである MLkit を用いて組み込んだ.

（※文責: 川村周也）

第 5 章　モデル構築

5.1　モデルの概要

モデルの目的は, アニメに関する情報をもとに, ユーザのアニメの好みを把握することである. 再検討により, アニメの特徴を表す情報と, 既存の評価情報を用いた行列補完をもちいることになった.

行列補完は疎行列に対し, 数少ない既存の要素から空になっている要素を尤もらしい予測値で補完する技術である. この技術は協調フィルタリングで用いられるような行列分解と次元削減, およびその再構成で行列を人工生成するものである. 成分数の多い疎行列を補完するには多くの行列計算が必要となる.

協調フィルタリングは, 購買やレビューなどの行動履歴をもとにユーザの嗜好や行動を, コンテンツを尺度として数値化する. その相関関係からコンテンツのレコメンドを行う代表的な手法である. (中間まで検討していた k 近傍法もレコメンドとして利用可能で, 協調フィルタリングに似た働きをする. しかし, k 近傍法はあらかじめそのラベルとなる変数をあたえた上で分類するアルゴリズムなので, 欠損値の推定は行われない.) その協調フィルタリングにも様々な手法を応用したものがあり, そのなかの手法に行列分解と次元削減というものがある.

ビッグデータを扱う解析には二つの大きな問題が起こる. ひとつは計算量の過多, もうひとつが「次元の呪い」と呼ばれる現象が起きてしまうことである. この「次元の呪い」は特徴量の過多による精度の悪化が発生を意味する. 行列分解と次元削減はこれらの問題を解消することができるため, 特徴抽出を大量に行う解析に不可欠な技術となっている. 本来 1 つの 2 次元行列 X であったユーザの行動履歴を, 2 つの行列 U, V の積で近似すると考える. このとき以下に示す式 5.1 の条件を満たす行をもつ行列 U, V を求める. これを行列分解という.

$$X \simeq U \times V^T \tag{5.1}$$

X を $m \times\ n$ の行動履歴を表す行列だとする. このとき 2 つの行列 U, V は $m > k > 0$, $n > k > 0$ を満たす定数 k を用いて, それぞれ大きさが $(m \times\ k)$, $(n \times\ k)$ となる. この k の値を小さくすることを次元削減という. 次元削減により, 再構成する際の計算量を減らし, 計算時間を大幅に短縮する効果がある. 再構成した行列の要素と元の行列の同じ要素を比較し, これを引数とした最適化関数を計算することで元行列への近似を行う. この再構成された行列は要素すべてが埋まった状態であり, この工程を行列補完という. 行列補完そのものがレコメンドシステムのモデルに利用されることは多々あり, その補完手法も多岐にわたる. 今回補完する疎行列は, 収集した既存のアニメ評価情報となる.

開発に当たって, 行列補完の試験実装用に NMF での補完, 本実装用に GC-MC[1] による補完を利用した. NMF は行列補完のなかでも実装が安易だったため, 行列補完モデルの試験実装に使用した.

再検討において GC-MC[1] を使うことになった主な理由は

- 行列補完による予測評価の生成がアプリケーションの趣旨に合致すること.
- アニメやユーザの特徴を付加情報としてデータセットに追加し, 予測に反映できること.

の 2 点である.

行列 X を $m =$ ユーザ数, $n =$ アイテム数, 要素をアイテムへの評価点数としたものとする. このとき必然的に U はユーザについての特徴行列, V をアイテムについての特徴行列となっている. これにより具体的な予測評価の生成を行うことが可能となる.

GC-MC[1] は 2017 年にその論文が発表され, これを実際に使用しているサービスはインターネット上で調査したところ見つからなかったため, 公開されたプログラムの応用が大きな課題となった.

以下に学習用のデータ整形および NMF と GC-MC[1] についての詳細を述べる.

（※文責: 坂元優也）

5.2 データセットの作成

5.2.1 スクレイピング

本アプリケーションの深層学習によるレコメンドモデルには, GC-MC[1] を用いた為, 複数種類のデータセットが必要となった.

一つ目が, 複数のユーザーによるアニメの評価情報であった. 数千単位の評価情報が必要になった為, ウェブスクレイピングを用いて収集することとした. ウェブスクレイピングとは, Web サイトから情報を抽出するコンピュータソフトウェア技術のことである. コーディングには Python を採用し, あにこれ [4] と呼ばれるアニメ作品を評価, 管理することができる Web サービスをスクレイピングすることとした. また, スクレイピングする際はあにこれ [4] に掲載されている全てのアニメ情報を収集するのではなく, 本アプリケーション内で管理されているアニメ作品のみとした. これらの評価情報は, 1 点から 5 点の 5 段階評価で表されており, 物語, 作画, 声優, 音楽, キャラ, この 5 種類の平均点の計 6 種類の点数によって構成されていた. 今回の深層学習モデルのデータセットには, ユーザーからアニメ作品への一意の評価点が必要であった為, 6 種類ある評価点の内, 平均点のみを用いることとした.

二つ目が, アニメの補助情報であった. こちらも評価情報と同様に, あにこれ [4] からウェブスクレイピングを行うことによって, 収集した. 今回のアニメの補助情報は, 制作会社, ジャンル, キャストの 3 種類を用いることとした.

三つ目が, ユーザーの補助情報であった. 本来の GC-MC[1] では, 評価をする側, 評価をされる側の両方補助情報が必要となった. しかし, 今回はデータセットを作成する際に, ユーザーの補助情報（性別や年齢など）を収集することができなかった為, ユーザーの補助情報を加味しないこととした. ユーザーの補助情報が収集できない原因は, あにこれ [4] で管理されているユーザーデータには, 性別や年齢の情報が付与されていなかったことであった.

上記の 3 種類の情報を保有している別の Web サービスや API を模索したが, あにこれ [4] 以上に情報が揃っている媒体が存在しなかった為, 本アプリケーションにはあにこれ [4] を採用した.

（※文責: 前田陸）

5.2.2 ジャンル推定

私たちは多くのデータをスクレイピングで取得し Firebase に保存した．このデータはモデルに組み込み, 1 つの指標として使う．さらに, アプリケーションでジャンル別にアニメを分類することにも使う．

しかし, そのデータが欠落している部分が多くあった．それがジャンルである．私たちが必要としていたジャンルデータが, 明らかに不足している．このジャンルデータがスクレイピングしてきたサイトには存在しなかった．この問題を解決するには, 手動で地道にジャンルに分ける方法, 他のサイトを探す方法, そして今回実施したジャンルを推定する方法の 3 つの手段が考えられた．我々はこの問題を解決するためにジャンル推定を行った．なぜなら, 手動で地道に分ける方法は, 多くの欠落と, 人員不足が伴うと感じたためである．さらに, 今回スクレイピングしてきたサイトと他のサイトとではジャンルの分別が異なる可能性があった．私たちが目指したのは全てのアニメを 25 種類のジャンルで分類することである．25 種類で分類する理由は, データ元である d アニメストアで使われているジャンルが 25 種類だったからである．

ジャンル推定は Doc2Vec を用いて, あらすじからジャンルを推定することにした．当初中間までは Word2Vec で試行錯誤していたが, 問題が発生し結局実現には至らなかった．Word2Vec では, 単語ごとに区切り, そのベクトルを足し算すれば文章のベクトルになるのではないかと考えていたが妥当なものにはならなかった．しかし, Doc2Vec の可能性に気づき, 問題を解決することができた．Word2Vec と Doc2Vec の違いは, 名前の通り, Word2Vec は単語の類似度を, Doc2Vec は文書の類似度をベクトルで表すというものである．Doc2Vec をどのように使うか説明する．

まず Doc2Vec を用いて各アニメのあらすじの類似度をベクトルで表した．さらにアニメのジャンルの類似度をベクトルで表し, あらすじの類似度とアニメのジャンルの類似度を最も近いものに対応させた．こうして, アニメのジャンルをあらすじから推定することに成功し, データセットの多くを補うことができた．実際には, ライブラリから numpy, gensim などを使用した．Doc2Vec への入力は単語毎に区切られている必要がある．このため, 形態素解析エンジン MeCab や，新語・固有名詞に強い NEologd と呼ばれる辞書を使用した．また, 日本語版 Wikipedia で学習した既存の Doc2Vec モデルを使用した．これらのオープンソースソフトウェアにより, 非常に簡単にジャンル推定を実現できた．

（※文責: 崎野也真人）

5.3 モデルの実装

5.3.1 NMF

NMF の説明の前に, 協調フィルタリングを用いたレコメンドについて具体例を用いて説明する．ユーザが 4 つのアイテムについてそれぞれ 1 5 点の評価をしたとき, 以下のようにベクトルとして表現できる．0 は未評価を表す．

$$user1 = (5, 0, 4, 4) \tag{5.2}$$

$$user2 = (4, 0, 0, 5) \tag{5.3}$$

$$user3 = (0, 5, 1, 0) \tag{5.4}$$

これらの多次元空間のベクトル同士がどの程度似ているかどうかを評価し, ユーザ 1 にレコメンドするべきアイテムを決める. ここで, ユーザの類似度を示すためにコサイン類似度を用いた.

$$similarity = cos(\theta) = \frac{A \cdot B}{||A||||B||} = \frac{\sum_(i = 1)^(n) A_i B_i}{\sqrt{(\sum_(i = 1)^(n) A_i)}\sqrt{(\sum_(i = 1)^(n) B_i)}} \tag{5.5}$$

計算の結果, ユーザ 1 と似ているのはユーザ 2 であることがわかった. つまりユーザ 2 が高い評価

表 5.1　user1 と他 user の類似度

	user 2	user 3
user 1	0. 8274288037841212	0. 10390486669322622

をつけたアイテムの中で, ユーザ 1 が未評価のアイテムをレコメンドすると良いと推測できる. ここで, 問題となるのがユーザ数とアイテム数が増えたときに発生する次元の増加である. データの次元数が大きすぎると, そのデータで表現できる組み合わせが飛躍的に多くなる. その結果, 手元にある有限なサンプルデータでは十分な学習結果が得られなくなる. そこで, 高次元データの特徴をできるだけ保持したままデータを低次元データに変換することでこの問題を解決する. この変換を次元削減と呼ぶ.

NMF は次元削減の 1 つである. 先程の例で考えると, (ユーザ数 × アイテム数) の行列 X と考えることができる. これを, ユーザ数 $> k > 0$ である k 次元に次元削減して変換し, ユーザの特徴を表す (ユーザ数 × k) の行列 U と (アイテム数 × k) の行列 V の積で近似する. 行列 X と行列 U は 0 以上の要素で構成される. 近似した行列 UV^T と行列 X を用いて行列 U と行列 V の値を学習する.

今回, 学習に用いたデータはユーザ数 3500, アイテム数 6500, 評価段階が 1 5 点からなるデータであった. 学習にはユークリッド距離の最小化を用いた. 性能評価のために, いくつかの評価済みのアイテムを 0 点に変更した. 学習後, 0 点に変更したアイテムの元の値と近似した値を比較することで性能を評価した. 評価の指標として, 2 乗平均誤差 (MSE) を用いた. 学習の結果, $MSE = 0.7941$ となった.

（※文責: 田渕知明）

5.3.2　GC-MC

今回, 本アプリケーションに採用した GC-MC[1] によるレコメンドシステム開発は, Python を用いて行われた. その際, 計 3 種類の GC-MC[1] モデルを開発することとなった. 3 種類のモデルは, 内部の行列演算処理は同様であったが, 開発に用いられたオープンソースの機械学習ライブラリがそれぞれ異なるものとなった. 使用した 3 種類の機械学習ライブラリは, Keras, PyTorch, TensorFlow であった. 3 種類の機械学習ライブラリを用いて, それぞれの GC-MC[1] モデルを構築した経緯を, 開発の流れに沿いながら説明する.

GC-MC[1] はモデル構造や内部演算処理が複雑であり, 論文を理解するのみでは開発が難しいものであった. 幸い, github 上に Python2 系で開発されたサンプルコードが存在した為, そのコードを参考に開発に取り掛かった. 今回の機械学習モデルは学習済みモデルを iOS アプリケーションに導入する想定で開発を行う必要があった. その為, まず初めにモバイルアプリケーションに機械

学習モデルを導入することのできるフレームワークの選定から行うこととした. フレームワークとは, プログラミングにおいて, アプリケーションプログラム等に必要な一般的な機能が, 予め実装された雛形のことである.

選定の結果, CoreML と呼ばれる機械学習フレームワークを用いることとした. CoreML は, Apple が提供している機械学習のフレームワークであり, 容易に機械学習モデル iOS アプリケーションに導入することができる. CoreML には, 大まかな特徴が 3 つ存在する.

- iOS デバイス上で学習済みモデルを実行することができる為, 外部との通信を必要としない.
- デバイスの CPU, GPU を活用するので最大限のパフォーマンスが発揮でき, デバイスの性能が上昇する.
- CoreMLTools を用いることで, Python の各種フレームワークで学習されたモデルを容易に CoreML フォーマットに変換することが可能である.

上記の特徴をふまえて CoreMLTools が対応している機械学習ライブラリが Keras のみであったことから, Keras を用いて開発を始めることとなった.

最初に着手したのは, スクレイピングによって収集したデータセットの整形であった. GC-MC[1] において必要となる情報はすでに明確化していた. よって, スクレイピングにて予め収集していたデータセットから以下の要素を作成した.

- ユーザ数
- アニメ数
- ユーザとアニメを合わせた総数
- クラス数 (今回は 5 段階評価であった為, 5 つのクラスとした)
- ユーザーの補助情報を表した行列
- アニメの補助情報を表した行列
- ユーザーとアニメを合わせた補助情報の総数
- 訓練用の隣接行列 (評価行列)
- 検証用の隣接行列 (評価行列)
- テスト用の隣接行列 (評価行列)

隣接行列とは, 有限グラフを表す際に使用される正方行列のことである. ただし, ここでの隣接行列はユーザー数×アニメ数×クラス数の　3 次元の行列として表現していた.

GitHub に公開されていたサンプルコードでは, TensorFlow と PyTorch によって開発されていたが, それぞれデータセットや使用用途によって, モデル構造に若干の違いが存在した. 今回の開発においても, サンプルコードを元に本アプリケーションに適したモデル構造で開発することとした. GC-MC[1] は, グラフ畳み込み層, 全結合層, 行列補完層の 3 種類のレイヤーによって構成されていた. 今回開発するモデル構造を図 5.1 に示す.

機械学習モデル構築の際に, 最初に行う作業であるモデル定義では, functional API を用いて行なった. functional API とは, 複数の入出力があるモデルや有向非巡回グラフ, 共有レイヤーを持ったモデルなどの複雑なモデルを定義するためのインターフェースである.

全結合層以外, 全てカスタムレイヤーであった為, TensorFlow の関数を用いて行列演算を行い, その全ての関数を Keras の Lambda 関数を用いてラップ処理を行なった. Python による機械学習モデルは, 入力値を変数や行列からテンソルに変換する必要があり, Keras ではテンソルのインスタンス化に Input 関数を用いた. GC-MC[1] には, 入力値が隣接行列, ユーザーの補助情報を表

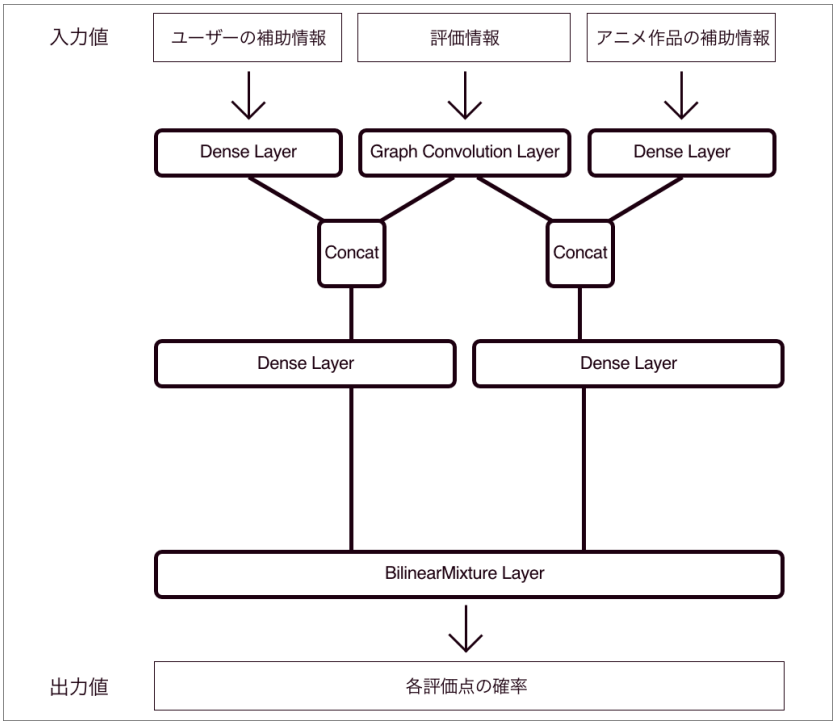

図 5.1　GC-MC のモデル図

した行列, アニメの補助情報を表した行列の 3 種類存在した.

入力値の次元数はそれぞれ違うものであった. モデルの定義, 構築, 学習までは順調に開発が進んだが, モデルを保存する際にエラーが発生した. 調査の結果, Keras のモデルは複数の入出力には対応しているが, その入出力値の次元数に差異がある場合には対応していないことが判明した. 改善策を模索したが, エラー解消には至らず, 期限も迫っていたことから, Keras による開発は断念することとなった.

しかし, 学習済みモデルを CoreML フォーマットに変換する CoreMLTools には, PyTorch はサポートされていなかった為, ONNX を経由する方法を選択した. ONNX とは, Open Neural Network eXchange の略であり, 深層学習モデルを様々なフレームワーク間で交換するためのフォーマットのことである. ONNX を使用することで, PyTorch で開発した学習済みモデルを CoreML フォーマットに変換することができた. しかし, PyTorch のモデルを ONNX へモデル変換する際に, 一部 ONNX がサポートしていない関数が存在した. ONNX にサポートされていない関数をサポートされている関数の組み合わせによって置き換えを試みたがエラー解消には至らなかった.

次に行なったのは, PyTorch によるモデル開発であった. Keras による開発の段階でモデル構造は決定しており, また, PyTorch はサンプルコードも存在した為, 開発は順調に進んだ. また, iOS アプリケーションへの導入には, Keras 同様 CoreML を用いることとした.

しかし, 学習済みモデルを CoreML フォーマットに変換する CoreMLTools には, PyTorch はサポートされていなかった為, ONNX を経由する方法を選択した. ONNX とは, Open Neural Network eXchange の略であり, 深層学習モデルを様々なフレームワーク間で交換するためのフォーマットのことである. ONNX を使用することで, PyTorch で開発した学習済みモデルを CoreML フォーマットに変換することができた. しかし, PyTorch のモデルを ONNX へモデル変換する際に, 一部 ONNX がサポートしていない関数が存在した. ONNX にサポートされていない

関数をサポートされている関数の組み合わせによって置き換えを試みたがエラー解消には至らなかった.

最後の手段として, TensorFlow による開発を試みた. 機械学習モデルのレイヤー構造は, Keras, PyTorch で開発したものと同様であった為, 開発箇所はレイヤー内部の行列演算処理のみであった.

開発は順調に進み, モデルの定義, 構築, 学習, 保存までは問題なく終了した. TensorFlow の開発では, iOS アプリケーションへ学習済みモデルをデプロイする際に, CoreML ではなく, TensorFlow Lite を用いた. TensorFlow Lite は, TensorFlow のモバイル環境向けの tool & runtime ライブラリ群であり, TensorFlow の学習済みモデルをモバイル環境で実行可能な形式に変換することができた.

TensorFlow Lite によるモバイルへの導入を開始したが, TensorFlow の学習済みモデルを TensorFlow Lite のフォーマットに変換する際に, サポートされていない関数が複数存在した. TensorFlow Lite 自体発表されて間もないライブラリであったことから, 複雑な行列演算関数をサポートしていなかったのであった. 幸い, サポートされている関数のみで同様の処理を書き直すことができた為, TensorFlow Lite フォーマットへの変換に成功した.

TensorFlow Lite のモデルは, 本来, Firebase 上に保存することで使用するのだが, 完成した GC-MC[1] のモデル容量が膨大であった為, Firebase 上に保存することが不可能であった. モデルを量子化することで上記の方法を可能にしようと試みたのだが, モデル内の行列演算処理の中に量子化に対応していない関数が存在した. その為, 量子化による保存を断念し, モデルを直接アプリケーションに実装することとした. レコメンド処理の際に, アプリケーションの動作の遅延が発生したが, 実装は成功した.

（※文責: 前田陸）

第 6 章　発表の評価

6.1　中間発表

6.1.1　ポスター作成と発表準備

プロジェクトの活動内容を 1 枚にまとめることが主な趣旨である．しかし作成上の問題は, 開発内容を全く知らない相手に対してどれだけ率直に理解してもらえるかということだった．本プロジェクトが開発するアプリケーションは, 個人間のレコメンドをサポートする未だかつてないコンセプトである．よってその意義や趣旨を理解させるポスターを作るためには, 相手に誤解をさせてしまう因子を徹底して排除しなくてはならなかった．コンセプトの意義はあくまで個人間のレコメンドに問題がある前提で意味を成すもので, それを納得させるための布石や, AI の利用法への疑問を感じさせないために, 担当教員からのアドバイスを多く受けた．開発計画に遅れが生じたことで期限間近に完成させ発表を行うことになった点は, 後期への課題となった．

（※文責: 坂元優也）

6.1.2　スライド作成と発表準備

中間発表前の時点では, 主に日常の問題の発見とその解決手法を探索して, 何が必要か, どんなことを実現したいかを提案し合った．スライドでは, それらについて理論を加えつつ順に沿って説明できるように留意した．

発表は現状の問題点の指摘から始めることにした．初めに具体的な状況を提示することで, 聴衆がイメージしやすくなると考えた．また, なるべく専門用語を避け平易な表現を心掛けた．続いて, 提示した問題点を解決するための手法の提案に移った．まず Deep Learning を用いること, それと連携したアプリケーションを開発することなど, 大まかな方針を示し, 次に各部の詳細を説明するようにした．Deep Learning を用いた学習アルゴリズムについては仕様が固まりきっていなかったため, すでにある程度要件が揃っていたアプリケーションの仕様の説明に焦点を絞った．アプリケーションの画面イメージを掲載し, どんな機能を持つか, どのような操作をするかなど, 直感的に理解しやすい構図になるよう配慮した．

一通り製作を終えたところで, 発表時間の確認, 台本の用意をし, 発表練習を行った．担当教員及び班員によるレビューを行い, 添削の参考とした．スライドのレビュー段階で, AI がユーザに対しレコメンドを行うシステムと混同される可能性が懸念された．そのため, レコメンドを支援することでユーザ間のコミュニケーションを促進する点を, プロジェクトの独自性としてアピールする考えにまとまった．アプリケーションを用いて相手との情報交換を行う部分を, 既存のサービスとの差別化であるとして強調した．

スライドの製作にあたって, それまでに取り組んでいた作業に区切りが見えず, 充分な製作期間がとれなかったことが反省点として挙げられた．

（※文責: 奥村拓馬）

6.1.3　中間発表に対する評価シートの内容

2019 年 7 月 19 日, 公立はこだて未来大学でプロジェクト学習の中間発表が行われた.

発表後, 聴講者に発表評価シートを記入してもらった. 発表評価シートに記入してもらう項目は 2 つある. 1 つ目は発表技術についてである. プロジェクトの内容を理解してもらうのに効果的な発表が行われているかを 1 点から 10 点の点数で観客に評価してもらった. 2 つ目が発表内容についてである. 発表技術についてと同様に, プロジェクトの目標設定と計画は十分なものかを 1 点から 10 点の点数で評価を受けるとともに, 発表技術, 発表内容それぞれについて評価の理由やアドバイスなどのコメントも記入してもらった.

中間発表で聴講者に記入していただいた発表技術, 発表内容それぞれについて評価の理由やアドバイスの一部を以下に示す.

- 身振り手振りが多くていい
- ポスターの図がわかりやすい
- スライドの情報量が少なく見やすい
- ジャンルが親しみやすい
- 解説がわかりやすい
- 両方のグループを観られるようにしてほしい
- スライドをと紙を見ながら話しているため伝わりづらい
- 下向いて話すのは好ましくない
- UI が出来上がっていて完成度が高い
- ゆっくり話してほしい
- アニメの前置きが長すぎる
- 目的, 手法が明確でよかった
- ディープラーニング必要か？

表 6.1 より発表技術は平均点が 7.1 点であった. コメントを見ると, ポスターの図やスライド, 解説がわかりやすいなどのコメントが多かった. 逆に, 前置きが長すぎる, 喋るのが早いことやスライドや紙を見ながらするのは好ましくないとのコメントが見られた. そのため, 反省点として, スライドを少なくし, ゆっくりと話せる時間的なゆとりを持つ必要と, 紙やスライドを見ずに発表をする練習の必要があると考える. 発表内容は平均点 7.6 点であった. コメントを見ると, UI が完成していたこと, 目的や手法が明確だったこと, ジャンルが親しみやすいなどのコメントがあった. また, 深層学習の必要性を疑問視するコメントもあった. しかし聴講者のほとんどにはプロジェクトの目的, 計画などの理解を得られたと考える.

（※文責: 崎野也真人）

表 6.1　中間発表の評価 (人)

評価	発表技術	発表内容
1	0	0
2	0	0
3	0	2
4	2	1
5	3	2
6	4	1
7	6	4
8	6	5
9	3	7
10	2	4
平均	7.1	7.6
合計	26	26

6.2　期末発表

6.2.1　ポスター作成と発表準備

ポスターの構成として概要, レコメンドシステムの実装, 結果と展望の 3 つに大別した. モデルの内容についてを中心に説明し, アプリケーションの概要についてはデモで説明するため省略した.

- 概要

  背景, 目標の項目で記述した. 背景では, 本グループが見出した問題点を分かりやすく伝えるため, 具体的な場面を挙げ, その結果どのような問題が生じるのかという展開で記述した. 目標では, 上記の背景で挙げた問題を解決するため AI で解決するということを記述した.

- レコメンドシステムの実装

  レコメンドシステムの概要, 予測評価データ, データ作成, モデルの概要の項目で記述した. レコメンドシステムの概要では, 具体的な場面を想定し, レコメンドの流れを順を追って説明した. その際, 視覚的に伝わるように説明文を補足するための図を掲載した. 予測評価データの項目では, 上記で説明した中で述べた予測評価データについて補足説明を行った. データ作成の項目では, レコメンドシステムで使用するデータの詳細について述べた. モデルの概要では, 利用した GC-MC[1] についての簡単な説明とともに, モデルの入力, 学習, 出力について述べた. また, レコメンドシステムの概要と同様に, 説明を補足する目的で図を掲載した.

- 結果と展望, 課題

  結果, 今後の展望, 課題の 3 つの項目で記述した. 結果では, GC-MC[1] を用いたモデルの精度について述べた. 今後の展望では, 今回のモデルの精度向上が見込めるであろう改善点について説明した. 課題では, 今回では, サービス自体の評価を行っていなかった点を指摘し, どのような実験を行えば, サービスの評価が行えるのか述べた.

（※文責: 岩成豪）

6.2.2　スライド作成と発表準備

製作期間は中間での発表準備期間よりも余裕があり, リハーサルを 2 回以上行えたこともあって, 前期で問題となった日程の課題は解決したと感じた. 期末の発表にはアプリケーションのデモを画面上で行う都合上, より簡潔に, しかしアプリケーションの仕組みがわかるように発表する必要がある. このプロジェクトが何を成し遂げたかということよりも, 成果物をどのようにして完成させたかを中心に発表スライドを作成した.

そのためモデルを詳細に説明することは最終発表で必要であると考えられた. しかし, この詳細説明は発表の質や聴講者の理解を著しく損ないかねないことがリハーサルで明らかになり, 担当教員からの指示も受け大幅に修正した. 発表の場は学生がほとんどであるため, どんな学生にも伝わるよう表現や内容は極端に簡略化した. なによりこのプロジェクトの成果物であるアプリケーションの魅力を感じてもらうことが発表で必要だと感じた. この目的に沿うよう, スライドはアプリケーションの仕様や使い方を大まかにとらえてもらえる内容に修正した. そのため, モデルやフロントの開発詳細については質問で受けることにした. 発表の重きを概要とし, 詳細を個人単位で受け付けることにより, わかりやすくアプリケーションの意義やコンセプト, 利用方法を説明できると考えた.

リハーサルにおける評価は大幅な修正を必要としないほどに悪くはなかった. 誤解を生まないため表現を調整するにとどまり, 時間や理解の易しさの問題を改善した. 全体を通して十分な発表準備ができたと感じている.

（※文責: 坂元優也）

6.2.3　期末発表に対する評価シートの内容

2019 年 12 月 6 日, 公立はこだて未来大学でプロジェクト学習の最終発表が行われた.

発表後, 聴講者に発表評価シートを記入してもらった. 発表評価シートに記入してもらう項目は 2 つある. 1 つ目は発表技術についてである. プロジェクトの内容を理解してもらうのに効果的な発表が行われているかを 1 点から 10 点の点数で評価してもらった. 2 つ目が発表内容についてである. 発表技術についてと同様に, プロジェクトの目標設定と計画は十分なものかを 1 点から 10 点の点数で評価を受けるとともに, 発表技術, 発表内容それぞれについて評価の理由やアドバイスなどのコメントも記入してもらった.

最終発表で聴講者に記入してもらった発表技術, 発表内容それぞれについて評価の理由やアドバイスの一部を以下に示す.

- 声が大きくて良い. アプリケーションを詳しく知れた
- (発表中) 下見ないといい. 説明わかりやすい. 需要ありそう. アプリケーションのクオリティ高い.
- 説明丁寧. 同時発表聞きにくい. ニーズがニッチすぎる.
- 聞きやすい. 試しに見て欲しい場合にいい.
- 聞きやすいが, グループで声がかぶる. 成果もわかりやすい.
- 資料よくまとめていた. 同時発表聞きにくい. 既存のレコメンドと変わらない.

- 理由と何をやるかわかりやすい．よくあるからいいアプリケーションだと思う．アニメ以外にも活用できそう．
- 聞きやすいが, グループで声がかぶる．成果わかりやすい．
- アプリケーションを作ったから結果どうなのか, 続きがないからわからない．下を見ている．隣で発表しているから聞きにくい．

表 6. 2 より, 発表技術は平均 7. 2 点であった．コメントを見ると声が大きくて聞きやすい, 説明が丁寧など好意的なコメントが多かった．逆に, A と B で発表を隣でした都合上, 声がかぶり聞き難かったというコメントもあった．そのため, 反省点として, もっと距離をとり, 十分なスペースを確保する, マイクなどを使い, より聞きやすくするなど対処が必要と考えられる．発表内容には平均 7. 3 点であった．コメントを見ると需要がありそう, アプリケーションのクオリティが高い, 成果がわかりやすいなど好意的なコメントがあった．逆に, ニーズがニッチすぎる, 既存のものと変わりないなど批判的なコメントもあった．そして, 作った結果どうなったのか, アニメ以外にも利用できそう, などのコメントもあり, 今後の展望への言及, 実験なども必要だったと感じた．一部批判的なコメントもあったが, 聴講者のほとんどには私たちが行なったプロジェクトの目的と計画に理解を得られた．

表 6.2　期末発表の評価 (人)

評価	発表技術	発表内容
1	0	0
2	0	1
3	0	1
4	1	0
5	1	1
6	7	1
7	4	6
8	7	6
9	2	3
10	2	4
平均	7.2	7.4
合計	24	23

(発表内容に関して, 1 人未記入者がいた)

（※文責: 崎野也真人）

6.3　総評

中間発表において, 課題だった聴講者の質問などに答えるために時間的なゆとりを持つことを A グループと B グループの同時発表によって解決することができたが, それにより別の問題が生まれた．2 つのグループが同時に発表することにより, 聴講者は聞き取りにくくなった．また, 台本を覚え発表することをできなかった．スライドを見ずに発表すること, ゆっくり話すなどは最終発表で

行うことができた. さらに, 中間発表と最終発表を通じて, 成果がわかりやすく聴講者から評価を得られアプリケーションのクオリティについても高い評価を得ることができた.

（※文責: 崎野也真人）

第 7 章　個人活動

7.1　川上達也（グループリーダー, Swift 班）

7.1.1　前期

前期では, グループリーダーに任命され, プロジェクトリーダーの補佐やグループを積極的に率いた. 主な活動としては, Swift 班として iOS アプリケーションの開発を行った. まず初めに Deep Learning を用いてユーザ間のアニメ情報を交換するアプリケーションについてどのような機能が必要かを明確にするために設計の要件定義を行った. その後, UI の要件定義を行い, その UI 案をもとに画面定義書を作成した. その後, Swift 班で役割分担をしたのち私は, 設定画面の作成を担当した. 私は Swift を利用したことがなかったため, 画面作成にあたってわからないところは随時 Swift 班の経験者や Web サイトを利用しながら作業を行っていった. しかし, 現段階ではアプリケーションの重要な機能面の担当ができるほど実力がないため, Swift の知識を習得する必要があると考えられる. また, アプリケーション開発だけではなく Python を用いて Deep Learning のモデル構築をしていく必要がある. したがって, 夏期休暇を利用して Swift の学習, Python のモデル構築経験を積むことと, 本プロジェクトで有効であるモデル案を論文や Web サイトを利用して検討をしていく必要がある.

後期では, まず, 前期で大まかな外装が完成したアプリケーションへ要素を増やしていく必要がある. 特に, Deep Learning 要素を前面に出せるようにアプリケーションの要素を検討していく必要がある. また, ユーザ間のレコメンドをしてくれる AI について夏期休暇を利用し, 論文を読み有用なモデル追実験を行った.

（※文責: 川上達也）

7.1.2　後期

後期は前期同様にグループリーダーとして, プロジェクトリーダーの補佐を行いながら, グループの進捗管理を行った. グループリーダーとしての主な活動は, まず, 夏期休暇の課題の進捗確認から行った, その後, 前期に話し合ったレコメンドシステムのモデルについて再検討を行った. 話し合いの結果, 試験的に NMF というモデルの構築を行い, NMF と並行で, GC-MC[1] というモデルの構築を行うこととした. それにり, 後期どのように活動していくかをプロジェクトメンバーと相談し, それぞれの役割分担と最終発表までのスケジュールを立てた. 話し合いの結果私は, Python 班として活動を行うこととなった.

Python 班としての主な活動は, 夏期休暇中にグループメンバー全員で集めた Firebase のデータのバックアップを取る作業から行った, まず, Firebase のデータを CSV 形式で保存することができるように, プログラムした. 次に万が一, データが破損, 削除された場合に備えて, データがいつでも挿入できるようにプログラムを行った. 次の活動として, モデルの補助情報に用いるためのデータの整形を行った. 夏期休暇中に全員で集めたデータの書き方が各々で異なっていたので, モデルの学習に使用するためにはデータの整形が必須であったためである. 整形したデータは Firebase

に登録されているデータの会社情報の整形を行った．まず, Firebase に保存されている会社情報を CSV 形式で取得し, Python で作成したプログラムと手作業を用いて会社名の表記の統一, 倒産, 合併した会社名を親会社名に統一, 正しくない会社名を正しい会社名に直す作業を行った．次の活動として, アニメのエピソード情報を Annict[3] の API を使用して Firebase に登録している 6876 タイトル分のアニメのエピソード情報を取得した．その後, 取得した情報を挿入するプログラムを用いて, Firebase にデータを挿入した．最後に, Python 班のメンバーが取得してきた, ジャンル情報を新たに Firebase に追加する作業を行った．その際, Firebase に登録されているアニメのジャンル情報が取得してきた情報に含まれていなかった場合は, 自分で調べて随時追加した．その後, 私は最終発表に向けて準備を行った．

最終発表では, 私は後半 3 回のスライドの発表を担当した．スライドで, プロジェクト全体の概要を説明した後, A グループの詳細発表をした．1 回目の発表は原稿を早く読み過ぎて時間配分がうまくいかなかったので, 2 回目以降は, 各スライドをゆっくり読み, 間を開けることで時間を調整した．今回の発表を通じて顔を上げて発表し, 必要に応じて身振り手振りをつけることが大切だと感じた．

プロジェクト全体を通じて, レコメンド支援アプリケーションを完成させることはできたが, 必要な作業を全て洗い出さずに作業を進めていってしまったことと, 経験者に大きなタスクを任せすぎたことが上げられる. そのため, メンバー間での作業量に差が出てしまった．解決策としては, 必要な作業を洗い出すことと, 経験者がサポートしながら複数人で行い, 未経験者のスキルアップを行いながら, 作業を行うのが良いと考える．また, 1 つの作業に時間をかけてしまい, 予定を延長するという問題もあった．そのため, 1 人で解決できなければ, 早めにグループメンバーや担当教員に相談して, 解決できるようにしていきたい．今後, このようなプロジェクト活動を行っていく際は, 今回の反省点を活かせるようにしていきたい．

（※文責: 川上達也）

7.2 川村周也（Swift 班）

7.2.1 前期

どのようなアプリケーションを作成するかの案出しにおいて, アプリケーションの基本的なアイデアを発案し, 要件定義, UI デザイン, フロントエンド開発を担当した．要件定義では班員と話し合いながらアプリケーションの機能を決定した．

UI デザインでは, 他アプリケーションなどを参考に Adobe 社の Illustrator を用いてデザインプロトタイプ, アプリケーション素材を作成した．フロントエンド開発では, Swift で iOS アプリケーションを作成した．開発では Git/GitHub を用いてソース管理を行いながら班員と分担して作業した．自分が主に担当したのは, モバイルアプリケーションのバックエンドサービスである Firebase を用いたログイン機能, モバイルデータベースである Realm を用いたホーム画面のプロフィール編集機能, iOS 端末間で P2P の Bluetooth 通信を行うフレームワークである MultipeerConnectivity を用いた交換機能, 検索画面において年代別にアニメを検索できる機能の実装である．前期の時点では, モデルとして有効なものができていなかったため, アニメの視聴管理機能を主として, 交換した視聴データから差分を表示させたり, アニメを効率的に検索できるアプリケーションとして開発した．以前から iOS アプリケーション開発を経験していたこともあり, 要件定義から画面定義, UI デザインに至るまで, スムーズに開発できた．

（※文責: 川村周也）

7.2.2 後期

夏期休暇にかけて, データ収集を行った. また, ジャンル推定のためにスクレイピングによって各ジャンルに属するアニメのリストを作成するコードを書いた.

後期は, 引き続きプロジェクトリーダーとして全体をまとめた. ML kit や Realm など, Android と iOS で同じライブラリを使う場面などは B グループにも情報を共有した. A グループでは, 主に Swift 班としてアプリケーション開発を担当した. アプリケーション開発では, アーキテクチャを変更し, コードの全面的なリファクタリングを行った. アーキテクチャを変更したため, 元のコードを参考にしながらも, 全ファイルに渡ってほとんど作り直しに近い作業となった. また, その過程で各画面に対応した Redux の State, Action, Reducer など, 同じような形式のファイルを大量に生成する必要があったので, メタプログラミングを導入した. メタプログラミングでは, Sourcery というライブラリを用いて, stencil という Swift のためのテンプレート言語を用いてボイラープレートコードを作成した. それによってコマンド 1 つで数ファイルが自動生成され, 作業効率が大幅に向上した.

また, 各アニメのサムネイル画像をネット上のサーバーから取得, 表示していたところを Firebase の機能の 1 つである, Firestore に保存, 表示する様に変更した. これに伴って RxDataSource というライブラリを導入し, Firebase から画像を取得し次第, 非同期的に画面に反映させるようにした.

GC-MC[1] の論文を読み, アルゴリズムの理解に努めた. できたモデルをアプリケーションに組み込む作業を担当した. 組み込みには, Firebase の ML kit を使用した. アプリケーション側では, Realm に保存されているユーザの評価データを多次元配列にして, バイナリ形式に変換し TFLite モデルに入力として与え, 出力として返された予測評価を Realm に再度保存するといった, 一連の処理を実装した.

リファクタリングと並行して UI デザインも改善した. データ収集によって表示できる情報量が増えたため, カード型からフラットデザインに変更し, なるべくシンプルになるようにした. また, 視聴管理のための登録 UI は片手での操作性の向上と縦スクロール時の視認性を阻害しないように, アニメーションするメニューボタンをデザイン, 実装した.

（※文責: 川村周也）

7.3 前田陸（Swift 班）

7.3.1 前期

6 月中旬まで参考書 [2] を用いて Python の学習を行った. 6 月下旬から Python 班と Swift 班に分かれ, 自分は Swift 班としてフロントエンドの開発を行った. 本プロジェクトでは, アニメのレコメンドや情報管理の機能を搭載したモバイルアプリケーションの開発が決まったため, まず初めに自分はデータベースの作成を担当した. データベースを作成するにあたって Firebase を用いた. データベースの作成が完了した後, Annict[3] が提供している API を用いてデータの収集を行い, Python モデル以外のバックエンドの設備を整えた. その後, 7 月に入ってからは本格的にアプリケーションの画面や各種機能の開発に取り掛かった. また, 開発を開始する前に自分はアプリケーションの画面定義書の雛形の作成を行った. その後, Swift 班のメンバーに各画面の定義書を作成

してもらい, 最終的に自分が全画面の添削を行い完成させた. 開発の段階では, Git/GitHub を用いてプログラム内のソース管理を行い作業を進めた. 自分が担当した箇所は, アニメ一覧画面, アニメ詳細画面, タイトルによるアニメの検索機能, Realm を用いたアニメ管理機能, ホーム画面による登録アニメの連携, Bluetooth 通信によって得られた結果の表示・反映であった. Swift 班のメンバーと協力して中間発表までにアプリケーションのプロトタイプを完成させた.

（※文責: 前田陸）

7.3.2 後期

中間発表終了後, 後期に行う作業の中で最も優先度の高いものを話し合った. その結果, アプリケーションのデータベース拡張兼深層学習モデルのデータセット作成のためのデータ収集が, 作業量が膨大であり最優先で行うべきものだと判明した. 夏休み終了後にこの作業を行なった場合, その他全てに遅延が発生してしまう可能性があった為, 夏休み中に取り掛かることとなった. 主に収集したデータは, データベースに登録されているアニメのあらすじ, キャスト, 監督, 製作会社の情報であった. 夏休み終了後, 本プロジェクトの要であるレコメンドシステムに採用する深層学習モデルの選定を行なった.

最初は難航したモデル選定であったが, 担当教員の助言により GC-MC[1] と呼ばれる手法に決定した. モデル選定後, 自分は GC-MC[1] に関する論文と GitHub 上に公開されていたサンプルコードを参考にデータセット作成を開始した. GC-MC[1] のデータセットには, ユーザーによるアニメへの評価情報とユーザーに関する補助情報, アニメに関する補助情報が必要であった. 夏休み中に収集したデータだけでは, アニメに関する補助情報しか無かったことから, Python を用いて Web スクレイピングを行い評価情報とユーザーの補助情報を収集した. その結果, およそ 20 万件の評価情報を収集できたが, その評価情報に対応するユーザーの補助情報を収集することができなかった. 幸い, ユーザーの補助情報が無くとも GC-MC[1] のモデルを構築し, 学習させることは可能であると判明したため, このまま開発は進むこととなった.

データセット作成が完了後, Swift 班として iOS アプリケーションの開発に復帰した. 自分は複数画面の UI と機能を修正した. Swift 班での作業がひと段落すると, Python 班によるモデル開発が難航していたため, GC-MC[1] のモデル開発を行うこととなった. 初めに, 班員と共に論文の読解から着手した. 複数の論文や GitHub 上のサンプルコードを元に, GC-MC[1] 内部の行列演算処理を理解した. その後, Keras, TensorFlow, PyTorch などの機械学習ライブラリを駆使し, GC-MC[1] のモデル開発を進めた. 完成した深層学習モデルは, 事前に作成していたデータセットを用いて学習を行なった. その学習済みモデルを iOS アプリケーションに組み込む必要があったため, Swift 班と協力しアプリケーションを完成させた.

（※文責: 前田陸）

7.4 田渕知明（Python 班）

7.4.1 前期

前期では, 主に Swift 班として iOS アプリケーション開発を行った. まず, メンバーが作成した画面定義を検討し, 画面定義書の作成を行った. その後, 誰がどの画面・機能を担当するのかを振り

分けた．担当した画面を実装するために不足している Swift や API に関する知識を, iOS アプリケーション開発経験者のメンバーを頼ったり, Web サイトを参考にしたりすることで補った．プロトタイプの開発終了の目処が立ったため, 7 月に Swift 班から Python 班へ異動した．

Python 班へ異動後は, Web サイトを用いてレコメンド AI に関する情報収集, 学習データの前処理を行った．学習データの前処理には Pandas のデータフレームを用いた．7 月に Python 班へ移動したため, 他の Python 班のメンバーと比べて知識量に差がある．そのため, 夏期休暇中に Python による機械学習について勉強をする必要があると考えた．具体的には, 東京大学の松尾研究室の公開しているコンテンツ, 機械学習系の参考書 [2] を使用した．また, 前期では Qiita などの Web サイトを主に学習時の参考にした．しかし, それでは知識の劣化コピーであるとアドバイスをいただいたので, 論文から知識を蓄えた．

（※文責: 田渕知明）

7.4.2 後期

後期では, 前期に引き続き Python 班として活動した．夏期休暇中の課題として, 学習に用いるデータの収集があった．メンバー全員で約 7000 件を収集した．収集するデータは, インターネット上にあるアニメのデータであった．Wikipedia 上のデータを, スクレイピングによる自動収集を試みたが, アニメごとに記載されている形式が異なっていたため断念した．そのため, 手作業でデータの収集を行った．しかし, 後期のプロジェクト学習初日までに終了しなかったため, 1 週間ほど, データの収集に費やした．収集したデータを用いて, データセットを作成した．そのデータセットを用いて, どのようなモデルでレコメンドシステムを実装するか検討した．まず, 簡単なモデルとして NMF(Nonnegative Matrix Factorization, 非負値行列因子分解) を用いた．このモデルは, 非負値行列を 2 つの非負値行列に分解するアルゴリズムで, もとの行列が持つ潜在的要素を明確にすることを目的としている．モデルの評価には, 2 乗平均誤差 (MSE) を用いた．行列の次元数, 学習率などを調整したが, 性能の向上が見込めなかったため, 他の手法を検討した．

2 つ目に実装したモデルは, GC-MC[1] という Deep Learning を用いたモデルである．このモデルは, NMF と同時進行で作成していたため, 作成途中から参加することとなった．主に行ったことは, プログラムのデバックとハイパーパラメータの調整である．インターネット上に公開されているプログラムを参考に実装したため, Python のバージョンによる不具合, 用意したデータセットの形式の不備などにより, プログラムが実行できないなどのバグがあった．Deep Learning のフレームワークとして, まず TensorFlow を用いた．モデルの完成後, iOS アプリケーションとして組み込むことができないことが発覚したため, 他のフレームワークである PyTorch, Keras を用いて実装したがどちらもアプリケーションに組み込むことができなかった．そのため, アプリケーションに作成したモデルを組み込むための技術調査を行った．その結果, TensorFlow Lite を用いることで, この問題を解決することがわかったため, Swift 班のメンバーがプログラムの変換を行った．

ハイパーパラメータを調整するために Optuna というツールを用いることを検討した．しかし, 学習時間が足りず Optuna を使うことができなかったため, 手動で実装することとなった．調整の結果, 大きな性能の向上はなかった．原因として, 手動による調整だったため, パラメータの調整が不十分であったためと考える．また, 必要とする入力データとしてユーザ情報があった．このデータはアプリケーションを使用しているユーザのデータから入手する予定であった．パラメータの調整時は, 内部の計算に影響を及ぼさないデータで代用していた．しかし, モデルの検証用にいくつかのユーザデータを用意し, パラメータを調整する必要があったと考える．その他のモデルも検討し

たが, 実装時間が足りなかったため断念した.

前期に技術力が足りないと感じたため, インターネット上に公開されているコードや, 書籍を用いて Deep Learning に関する学習をした. しかし, 前期の課題であるデータ収集に時間を取られてしまい, 思ったように学習が進まなかった. 後期が始まってからは, TensorFlow のチュートリアル等を使いスキルアップを図った. しかし, レコメンド用のモデルの構築の進捗が振るわなかったため中断した.

プロジェクト全体を通して感じたことは, 全体的に事前準備不足であったことである. 特にモデルの作成において, どのフレームワークを使用するかよく検討する必要があったと考える. 使用するモデルを検討してからデータ収集を行えば, 不要となるデータ収集の時間を削減できたためである. また, モデルの開発を複数人で行ったが, 比較するモデルの開発のことを考えて人数を割り振るべきだと感じた.

（※文責: 田渕知明）

7.5 崎野也真人 (Python 班)

7.5.1 前期

参考書 [2] で学習し, 解説等を順番に行った. また, GPU を使いニューラルネットワークの学習を実行し, 実際に結果を得た. 次に画面定義書に従い担当に分けてアプリケーション画面の作成をした. アプリケーションのホーム画面の作成補助をした. iOS アプリケーションに接続するサーバを Firebase にしたため, Firebase を 0 からメールアドレスとパスワードの登録とログイン機能を実行できるようにした. 中間発表でスライドを用いて発表を行った. それに対するレビューの集計, 主なコメントから改善点を探った. 7 月に Python 班に異動し, 知識と経験が不足しているためにそれを補えるよう夏期休暇中に k 近傍法, 分類手法などの先行研究論文を確認したい.

（※文責: 崎野也真人）

7.5.2 後期

Python 班として活動をした. メンバーとともに夏期休暇から大量のデータ収集を行った. 主に収集したデータは, データベースに登録されているアニメのあらすじ, キャスト, 監督, 制作会社の情報であった. また, そのデータを Firebase に入れた. そのデータをチェックして大量のデータの整形と不足しているデータを補った. 手動でのデータ入力によって完成したデータセットが, 手動故にミスや, 一貫した表現のなされていないものを共通した単語に入れ替えてプログラムで使用できうるものにした. 例えば, アニメの会社や声優などで, ”・” や”, ”など, 前後にスペースや半角と全角が混ざっているなどをプログラムでの仕様に耐えうるように綺麗に整形した. データ量が膨大なため, これを整形するためのプログラムコードを様々に実験しながら書いた. さらに, 他のプログラムに使用するための形式や他の CSV ファイルと結合などのためのプログラムを作成し CSV ファイルの入力や出力を大量に行った. この作業はモデル作成の際に必要なデータセットを綺麗にすることであり, 非常に重要な役割だったと考えている. .

そして, ジャンル推定のために必要な手法について Web で情報収集した. その結果, 機械学習を用いた手法が有効だと感じた. Word2Vec と Doc2Vec の違いもわからない状態だったため多くを

学習する必要があったが, Doc2Vec を使用し, 既存の学習モデル, ライブラリなどを使用して非常に簡単に実装することができた. その結果, あらすじが有り, ジャンルのないアニメについては全てプログラムを実行してそのジャンルを埋めることができた.

（※文責: 崎野也真人）

7.6 坂元優也（Python 班）

7.6.1 前期

Python による教師あり学習の概要を学習したのち, Python によるレコメンデーション機能の実装にむけてアルゴリズムを調査した. その後, 機械学習の手法を基幹にアルゴリズムを設計することにした. データ整形による時間の浪費が激しかったが, グループメンバーの手を借り解決へ向かいつつあった. 前期時点では, 後期にむけてアルゴリズムの洗練に専念するとともに, 個人間のレコメンデーションの内容を生成するためのアルゴリズムを作成する予定であった. Python による先行研究を参考にするため, 言語として Python を学習するとともに, k 近傍法以外の分類アルゴリズムやクラスタリングについても知識を深める必要があると感じていた.

（※文責: 坂元優也）

7.6.2 後期

前期で不完全な見通しで始まってしまったアプリケーション開発だったが, 後期最序盤にデータベースの基礎が完成した. そのため, 早い時期にレコメンド用のアルゴリズムの構築に取り掛かることができた. アルゴリズムについて再検討の末, 担当教員からの助言により実装への道筋が見えるようになったことがこのプロジェクトに大きなタスクの変化だと感じた.

ユーザやアニメの特徴ベクトル化という観点から離れ, 既存のレコメンドシステムに使われるような行列補完の応用が用いられることになった. この時使用した GC-MC[1] モデルは, 参考にした論文の著者自らが GitHub 上に配布したプログラムもあったのだが, 前期で利用した Keras のような機械学習用のライブラリで再現できないような複雑な計算を含むものであった. その解読は私にとって困難を極めたため, その実用化に大きくは貢献出来なかった.

TensorFlow による行列計算を幾度となく行うこのプログラムは, その教師ありデータ部分の予測値の計算を行い精度や損失値を算出することができたが, 肝心な教師なしデータ部分の予測値を算出することが不可能であった. 大きな理由として, TensorFlow は 1 つのテンソルにして計算を始めてしまえば, その要素を取り出して個々として扱う計算はできない. そのためには 1 つひとつを list に埋め込むなどの余分な作業が必要であった.

しかし約 6000 × 3000 の要素を持つ行列ですら膨大な計算時間, メモリが必要になってしまうことがわかったため現実的ではないこともわかった. アプリケーションのユーザから得られる新しい情報や, それに付随する情報を加えて再び学習させるなら, なおさら大量の資源が必要になってしまうことが考えられた. このため第 3 者が提供した PyTorch 版の GC-MC[1] プログラムを応用することになり, 紆余曲折を経てこのモデルを理解する役割を外れた. これ以降のモデル関連は前田, 田渕に大部分を任せることになり, 不甲斐なさを感じた. しかしこのモデルの解読で得た経験はこれからのプログラミング, さらにはアルゴリズムの理解に生かすことができるだろうと感じている.

その後の発表準備では前期に比べて余裕ができ, 生徒向けを意識した理解しやすい発表を行うことができたと感じている. プロジェクトを通して仲間とアプリケーション開発をできた, この貴重な経験を与えてくれたプロジェクトメンバー, 担当教員の方々に感謝する.

（※文責: 坂元優也）

7.7 奥村拓馬（Python 班）

7.7.1 前期

まず, Python の環境構築に始まり, 基本的なふるまいや構文の理解, 各種ライブラリの導入や利用を行った. 成果物の目標が定まってからは, 実現のためのモデルについて情報収集, ウェブスクレイピングの知識理解とプログラミングに注力した. 今後の課題や展望には以下の工程が挙げられた.

- データ整形の形式を確定し, 実現
- 単語分割
- 出力の考察
- 膨大なデータを収集して適用
- アプリと連携, 採用する部分を決定

（※文責: 奥村拓馬）

7.7.2 後期

後期の開始前より, 学習に必要となるデータの収集を行うこととなった. それは, Firebase 上に登録されたアニメに対し, 監督, 制作会社, あらすじ, キャストのデータを Web 上から収集するというものであった. 後期の数週まではその作業を進めた.

その後は, 主に取得したアニメに対するジャンル情報を付加する作業にあたった. まず初めに検討した手法は, d アニメストアに登録されたジャンル情報を利用するものであった. d アニメストアに登録された各アニメには, ジャンル情報が対応付けられている. そこで, Firebase 上に登録したアニメタイトルを d アニメストアにあるタイトルと比較して, それらが一致した場合に後者と対応するジャンル情報を登録するという分類を試みた. このとき, d アニメストアよりタイトルを json 形式で保存してジャンルごとに管理し, プログラムで読み込む方式を採用した. ジャンル数は d アニメストアをもとにしつつ調整した結果全 13 種とし, また各々に登録されたタイトルが完全一致した場合に分類済みとした.

この条件により分類済みとなったタイトルは約 6800 タイトル中 3400 程度であった. 未分類となったものについて, タイトルの部分一致でも分類するようにして表記ゆれに対応させたり, 入力した任意の同じ文字列を含むタイトルに対して, まとめて同じジャンル情報をつけるプログラムを組むなどして対処した. その他, 使用されやすい単語を含むタイトル (魔法, 戦隊など) に対し一括振り分けを行い, すでにジャンルが登録されているタイトルから他の未分類のタイトルのジャンルを推定させるなどした.

ただし, これらの修正は根本的な解決には至らなかった. 未分類となったアニメは, タイトルだけでジャンルを推定することが困難であったり, 頻出するような単語をタイトルに含んでいないことが多かったためである. 視聴情報も乏しく, アニメに知識のある班員であっても手作業かつ迅速に

振り分けるのは至難の業であった．そのため, 未分類のものは, 全てを「その他」としておき, 期限内に順次見分けていくことが現実的であると, 一時妥協案を採った．最終的に, Doc2Vec を用いてアニメのあらすじからジャンルを推定する手法を班員から得て, それを用いて未分類だったアニメ全てに対するジャンルの振り分けを行った．この作業の前後段階で, 目視によってジャンルが正しく振り分けられているか確認した．

（※文責: 奥村拓馬）

7.8 岩成豪（Python 班）

7.8.1 前期

プロジェクト開始後に, ブレインストーミングでアイデア出しを行った．アイデアが決定した後, Python のフレームワークである Keras を用いて, Deep Learning の勉強をした．その後, アニメのあらすじをベクトル化するために, 情報収集をした．文章のベクトル化に Doc2Vec が適していると判断し, 試験的にアニメ情報サイトであるあにこれ [4] からウェブスクレイピングで得たあらすじ情報を元に, あらすじのベクトル化を実施した．結果, 懸念される点が浮かび上がった．それは, 文章の長さが統一されてないために, 文章間の類似度がうまく計算されていないのではないかということである．よって, データ整形をする際に, あらかじめ文章の長さを統一する必要があるのではないかと考えた．

前期を振り返って, Deep Learning の基礎が理解できた．特に文章に対する学習について理解を深めることが出来た．後期では, 実際に本プロジェクトのサービスで使用するデータセットでベクトル化を行うので, Doc2Vec の仕組みについて改めて調査をすると共に, 別の文章のベクトル化手法について調査していきたい．また, プログラミングスキルや, その他情報共有するためのツールに対する知識が不足していると感じたため, Python や GitHub の学習をしていきたい．

（※文責: 岩成豪）

7.8.2 後期

後期に入る前に, 夏期休暇中に, 開発するレコメンド機能に必要なデータ収集を行った．具体的には, Firebase のデータベース上に, アニメ情報 (制作会社, 監督, 年度, あらすじなど) を公式サイトや, Wikipedia から収集, 整理する作業を行った．後期が始まった後は, レコメンドシステムについての先行研究の調査を行った．その際に, メンバーが発見した GC-MC[1] を元にレコメンド支援システムを開発することが決定した．先行研究の調査終了後は, 主に GC-MC[1] について理解するための学習, および GC-MC[1] に関連すると思われる VGAE[9] 等について調査した．その後, Firebase 上のデータに加えて必要になったデータの前処理を行った．具体的には, データの形式を json ファイルにまとめるなどした．その後, ポスターの作成を担当し, 各メンバーに対して, 再度, モデルの構成, サービスの目的などの確認を行った．

後期の活動を振り返って反省すべき点がいくつかある．まず, 前期で挙げた Python の学習を夏期休暇中にほとんど行っていなかったことである．結果, 後期でプログラミングを行う際に, 非常に苦労した．次に, Deep Learning, およびそれに関連する数理的知識の欠如である．GC-MC[1] を理解するためには, 上記のより高度な知識が必要不可欠であったが学習不足であったため, 最終発

表会を迎えるまで理解するに至らなかった. よって, これらの知識を夏期休暇中に蓄えておくべきだったと考える. 次に, 積極性の欠如である. ポスターの担当を任された段階では, 本グループが開発するサービスの目的, モデルの概要などほとんど理解できていなかった. よって, ポスター制作にも必要以上に時間がかかった. 後期開始時から, 随時各メンバーとの情報共有を積極的に行っていればよりスムーズにポスター制作が行えたと考える.

プロジェクト全体を通して, もっと積極的にサービス開発に携わるべきだった. 受け身であった結果, 技術的な面でグループにほとんど貢献出来なかった. 今後, グループ活動を行う際は, 今回の経験を糧に積極的にメンバーとコミュニケーションをとって参加するようにしたい.

（※文責: 岩成豪）

第 8 章　まとめ

8.1　前期のまとめ

まず中間目標として, 中間発表までにモバイルアプリケーションのフロントエンドの開発を行い, プロトタイプが動かせる状態まで作成することと, Word2Vec, Doc2Vec, k 近傍法による分類化の 3 つの手法を使いプロトタイプモデルを作成することを決定した. その後, Swift 班と Python 班の 2 つに分かれ, それぞれのグループで活動を行っていった.

Swift 班は, アプリケーションのフロントエンド開発のためにどのような機能が必要か話し合った. 話し合いの結果, 今回作成するアプリケーションは, アニメのレコメンドや情報管理, 対面によるレコメンド支援機能が必要であったので, これらの機能を開発することに決定した. その後, アプリケーションの画面定義を行い, メンバー間で画面の UI と機能の意識統一をした. 開発では, Git/GitHub を用いることで, プログラム内のソースコードの管理を行い, 効率的に各種機能の作成を行っていった.

前期の活動を振り返ると, 目標としていたアプリケーションのフロントエンドの完成は遂行した. しかし, Swift 班の中での仕事量に偏りがあったと感じる. 理由としては, 基礎学習の時間を取れなかったため, アプリケーション開発経験者が中心になって開発を行い, 重要な機能やデバッグを任せたことが考えられる. したがって, 後期の活動に向けて, 夏期休暇中に各自学習をし, 独力で問題解決ができるようになる必要がある.

Python 班ではレコメンド機能の実装を目的とし, アニメのデータ収集, 3 つの手法を検討した. データ収集にはウェブスクレイピング, Annict[3] が提供している API を用いた. 抽出したデータの偏りや, 不要な情報を削除するなどの処理を行い, 学習に悪影響となる要素を取り除いた. 現在検討している手法は, Word2Vec によるアニメレビュー解析, Doc2Vec によるアニメのあらすじ解析, k 近傍法によるユーザやアニメの分類表現である. どの手法も実装できていないため, 夏期休暇と後期に取り組む予定である.

課題として, 開発に計画性がなかったため, 中間発表間際に発表資料を作成したことが挙げられた. また, 技術不足による, タスクの極端な偏りがあった. 後期ではそういったことが無いよう, 全員で開発計画・進捗を確認し, タスクを分散させることにした.

（※文責: 川上達也）

8.2　後期のまとめ

夏期休暇中に, アプリケーションとモデルに使用するアニメ情報の収集を行った. また, Python 班が前期中に検討していた手法では, 本プロジェクトで行うレコメンド支援がうまく進まないという結論に至った. そのため, 後期開始時に Swift 班と Python 班の 2 グループで, 目標の再設定を行った.

Swift 班は, まず前期中に完成させていたアプリケーションの見直しを行った. 見直しの結果, 大きく 3 つの改善点が見つかった.

- Bluetooth 通信を行なっている際にアプリケーションがクラッシュする.
- アプリケーションのプログラムの構造が複雑になりすぎて, 一つひとつの処理が非常に重くなる.
- アプリケーションの UI が分かりづらく使用しにくい機能, ボタンがある.

これらを改善することと, レコメンドシステムのモデルを組み込めるようにアプリケーションの本実装を行うことが Swift 班の後期の目標とした. アプリケーションの改善のために Swift 班はまず, 使用技術選定を再度行った. (選定した技術の名前を入れるのと若干の説明). 開発では, 前期同様に, Git/GitHub を用いる ことで, プログラム内のソースコードの管理を行い, 効率的に各種機能の作成を行っていった. 開発の流れは, アプリケーションのリファクタリングを行い, それと並行して UI の変更も行っていった. リファクタリングが終了した後, アプリケーションの本実装に取り組んでいった.

Python 班は, レコメンドシステムに用いる技術の選定を行った. その中で, 行列補完という技術を用いれば本プロジェクトで行うレコメンド支援のモデルが作れるという結論に至った. そこで, Python 班は試験的に, NMF によるレコメンドシステムの実装を行い, 行列補完の有用性の評価を行った. また, NMF と並行で, GC-MC[1] というモデルの実装を後期の目標とした. また, Python 班をデータ収集, 整形を行うグループとモデルの実装を行うグループに分けた.

データ収集, 整形グループの活動としては, まず, 夏期休暇中に全員で Firebase に登録した情報のデータ整形を行った. 次に, GC-MC[1] の学習に使用する情報を収集, 整形する活動を行った. 最後に, アニメのエピソード情報を Anniict[3] から取得して Firebase に登録する作業を行った. モデルの実装を行うグループは, まず, NMF のモデルの構築を行い, どれだけレコメンドできるかの評価を行った. その後, GC-MC[1] の構築を行った. インターネット上に公開されている実装が複数あったため, それらを参考にしながら, Keras, TensorFlow, PyTorch の 3 つのライブラリでそれぞれ実装を行った. しかし, モデルの不具合や, アプリケーションに組み込めない問題が発生した. その後, TensorFlow Lite で実装することで問題解決に至った. その後, パラメータ調整に取り掛かった. しかし, 時間的な都合により, パラメータの自動調整を行う Optuna を組み込めなかったため, パラメータの調整を手動で行った. そのため, パラーメタの調整が不十分であったと考える. 今後, パラメータの自動調整を行い, 精度を向上させる必要がある.

今後の課題として, 今回構築した, レコメンドシステムが結果を出力するまでに非常に時間を有するモデルであった. そのため, 対面による Bluetooth 通信での交換は, 非常に非効率である可能性が挙げられる, また, 事前準備不足やスケジュール管理がうまくいっていなかったことによる時間不足でチューニング作業や評価を行う時間がなかったことが挙げられる. したがって, これらの問題を解消しプロジェクトを成功させるためにも, 事前準備やスケジュール管理が非常に重要であると考える.

通年の活動を通じて, Deep Learning に対する知見やレコメンドシステムに関する知識, チームで連携をして 1 つの物を作り上げる難しさなどを学ぶことができ, 非常に有意義なプロジェクト学習にすることができた.

（※文責: 川上達也）

参考文献

[1] Rianne van den Berg, Thomas N. Kipf, Thomas N. Kipf, Graph Convolutional Matrix Completion (2017)

[2] Python と Keras によるディープラーニング
Francois Chollet (著), 巣籠 悠輔 (訳), 株式会社クイープ (翻訳), マイナビ出版, 2018 年

[3] Annict, 2019 年 7 月 17 日 アクセス, [online] https://annict.jp/

[4] あにこれ, 2019 年 7 月 1 日 アクセス, [online] https://www.anikore.jp

[5] WATCHA, 2019 年 7 月 1 日 アクセス, [online] https://watcha.com

[6] kaggle, 2019 年 7 月 1 日 アクセス, [online] https://www.kaggle.com/tanetboss/user-clustering-for-anime-recommendation

[7] scikit-learn, 2019 年 6 月 24 日 アクセス, [online] https://scikit-learn.org/stable/modules/neighbors.html

[8] Redux+Rx を活用した iOS アプリアーキテクチャ, 2020 年 1 月 27 日アクセス, [online] https://qiita.com/susieyy/items/23d44f28c6a6915c58e2

[9] Thomas N. Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. NIPS Bayesian Deep Learning Workshop (2016)