AIするディープラーニング

AI love Deep Learning

川村周也 Shuya Kawamura

1 背景

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. また, 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考えられるようになった. 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 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. 深層学習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, この深層学習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.

2 基礎知識の習得とテーマ決定

4 月から 5 月にかけて, 勉強会を行い, 参考書 [1] を用いて DeepLearning の仕組み・原理を実際に実装しながら学んだ. 事前に参考書の章を一つずつ各自自習し, 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. 6 月の初めに, 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, 最も有用な意見を話し合い, 本プロジェクトで行う活動の内容を「アニメ作品のレコメンド支援」「感情分析を用いたネット小説のオススメ」の 2 つに絞った.

3 課題解決のプロセスとその内容

3.1 グループ A: アニメ作品のレコメンド支援

3.1.1 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. Annict[2], あにこれ [3] 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[4] 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.

3.1.2 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.

3.1.3 目標設定

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, グループの目標として

- 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.

3.1.4 アプリケーション開発

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[2] 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[5] 上にアニメのデータベースを作成した. そして Firebase[5] 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 TensorFlow Lite[6] 形式のモデルを Firebase[5] の機能の 1 つである MLkit を用いて組み込んだ.

3.1.5 モデル開発

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与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することを意味する. 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 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.

$$X \simeq U \times V^T \tag{1}$$

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(Non-negative Matrix Factorization)[7] での補完, 本実装用に GC-MC(graph convolutional matrix completion)[8] による補完を利用した. NMF[7] 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[8] 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.

3.2 グループ B: 感情分析を用いたネット小説のオススメ

3.2.1 該当分野の現状・従来例

現在，インターネット上にはさまざまなレコメンデーションサービスが存在する．例えば，Google の

「Google 広告」や Amazon の「あなたへのおすすめ」などの Web サービスや，個人的に作られた Web サイトがある．Web サービスでは，今までに購入したものや閲覧記録などをもとにユーザが興味を持ちそうなものをピックアップしてオススメする．また，個人的に作られた Web サイトでは，そのサイトの作成者個人が気に入った商品をオススメしたり，ジャンルごとに分かれたものから最も人気のものをオススメしたりする．

3.2.2 現状における問題点

オススメの対象を一般的な小説に限定すると，これをオススメする Web サービスはいくつか存在する．しかし，ジャンルにとらわれていたり，オススメ内に過去に読んだことのある作品が紛れ込んでいたりする．さらに対象をネット小説にまで限定すると，これに関するアプリケーションは電子書籍リーダーしかなく，オススメ機能は備わっていない．ネット小説とは，一般的にインターネット上の小説投稿サイトで公開されている小説であるが，ここでは「小説家になろう [9]」という小説投稿サイトの小説とする．

3.2.3 本グループにおける目的

本グループの目的は 2 つある．1 つ目は，ネット小説の内容を感情分析し，ネット小説をオススメするシステムを開発することである．内容を分析することで，従来のオススメよりもパーソナライズされたオススメが可能になるはずである．2 つ目は，この機能を Android アプリケーション（以下，アプリケーションと表記）として実装することである．アプリケーション化することによって，このシステムをより簡単に利用できるようになるはずである．ネット小説の感情分析およびネット小説のオススメには Deep Learning を用いる．ここでの感情分析とは，ネット小説の本文の単語と感情を表す単語との関連性をそれぞれ数値化することである．感情分析は，既存のモデルである Google の「Word2vec[10]」で行い，ネット小説のオススメは，CNN（Convolutional Neural Network）で行う．

3.2.4 手法の検討

このテーマを踏まえて，各自で感情分析の手法を調べ試行した．まずデータを取得するため，Web スクレイピングを用いてネット小説の文章を取得した．この Web スクレイピングで，「なろう API」が最も優れた手法であると判明した．また，試した手法は「Word2vec」と単語の辞書をベースとしたもの，IBM の API である「Watson Tone Analyzer」の 2 つである．その中で，「Watson Tone Analyzer」は日本語に対応したものがなく，これは利用しないことにした．

3.2.5 本文の感情分析

Word2vec では，2 つの単語間の類似度を求められる．これで求められる類似度は，1.0 に近づくほど単語同士が類似しており，-1.0 に近づくほど単語同士が類似していないことを意味する．まずはこの Word2vec の機能を用いて，学習させた全ての単語と 4 感情との類似度を求めた．以下に単語と感情との類似度の計算結果の例を示す．

表 1: 単語と 4 感情との類似度計算のイメージ

	涙	笑顔	激怒
喜び	0.1	0.9	-0.7
怒り	-0.2	0.1	0.8
哀しみ	0.8	-0.6	-0.5
楽しみ	0.2	0.7	0.3

その後，感情ごとに単語と類似度の組み合わせをリスト化し，これを類似度リストとした．小説の本文は，まず全ての話を結合して 1 つのネット小説に対して 1 つのテキストファイルにした．その後そのテキストを分かち書きしネット小説を単語リストに編集した．

次に，単語リスト内の単語を 1 つずつ 4 感情それぞれの類似度リストと照合することでネット小説の単語リストを類似度を表す数値リストに変換した．これが感情分析の具体的な流れである．ただし感情分析の際に，ネット小説に含まれる全単語のうち Word2vec が学習していない単語が 5 %程度あった．これは今回使用せず，残りの約 95 %の単語を感情分析に用いた．

3.2.6 判別用モデルの作成

ユーザにネット小説を的確にオススメするために，オススメの判別用モデルを作成した．入力は小説データの元となる感情データとユーザデータの元となる感情データとした．また，出力はユーザに入力した小説をオススメすべきかそうでないかの判定とした．このモデルでは，実際に「小説家になろう」のユーザ（以下，なろうユーザ）がネット小説を評価した情報を学習させた．判別用モデルの入出力を図 1 に示す．

小説家になろうでは，ネット小説を文章とストーリーのそれぞれの観点から最大 5 ポイント，合計 10 ポイントで評価できる．このデータをなろうユーザ 100,000 人分収集した．データセットのユーザデータの元となる感情データは，ユーザが高い評価をつけ

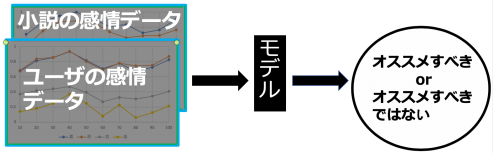

図 1: 判別用モデルの入出力

たネット小説の感情データを平均して作成した．小説データの元となる感情データはデータベースに保存されているものとした．ラベル付けには評価ポイントを用いた．最大 10 点の評価ポイントで，10 点をつけたネット小説をオススメすべきと，それ以外をオススメすべきでないとラベル付けした．

判別用モデルは，2 つの感情データを入力とする 2 分類問題である．また，入力の形式が画像の入力形式と同様に 3 次元の配列になっている．そのため，全結合層の他に畳み込み層を組み込んだ CNN を構築した．モデルを構築したあと，モデルでの最適なハイパーパラメータを求めるためにグリッドサーチを行った．その結果，正解率が 70 %程度のモデルが完成した．

3.2.7　アプリケーションの概要

私達が開発したアプリケーションは，ネット小説を感情分析した結果をもとにユーザにネット小説をオススメするアプリケーションである．このアプリケーションは，約 3,500 件のネット小説のデータが登録されており，それぞれのネット小説の感情グラフを作品名や作者名，あらすじなどとともに見ることができる．また，好きな本や嫌いな本の登録を行うことで，あなたへのオススメやユーザグラフが表示されるようになる．

アプリケーションの名前は，感情分析を使って本をオススメするアプリケーションであるため，感情を意味する英単語である「emotion」と本を意味する「book」の 2 つの単語を組み合わせた「EmoBooks」とした．

4　まとめ

本プロジェクトでは Deep Learning を用いて, 日常生活における問題解決のためのモバイルアプリケーションを開発した．A グループは, 個人間でアニメ作品をレコメンドする際, 相手ユーザにレコメンドすべき作品を提案する iOS アプリケーションを開発した．B グループは, 小説の本文の感情の推移を元に, ネット小説のレコメンドを行う Android アプリケーションを開発した．両グループともモバイルアプリケーションは完成し, 発表会ではデモを用いて聴講者に実際にアプリケーションを触ってもらうことができた．実際にリリースして欲しいなどの好意的な感想をもらうことができた．

5　今後の課題

A グループでは, 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．

B グループでは, オススメのモデルを最後まで作り上げアプリケーションに組み込むこと，オススメのシステムの評価実験を細かく行い改良を行うことが挙げられる．オススメのシステムの改良としては，ニューラルネットワークの層の数やネットワークの種類など，試していないものが沢山あるので，さらにいくつかのモデルを作り試してみることが必要である．また，そのモデルをアプリケーションに組み込みアプリケーションの評価も行う必要もある．

参考文献

[1] 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

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

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

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

[5] firebase, 2020 年 1 月 21 日アクセス, [online] https://firebase.google.com/?hl=ja

[6] TensorFlow Lite, 2020 年 1 月 21 日アクセス, [online] https://www.tensorflow.org/lite?hl=ja

[7] scikit-learn で Non-negative Matrix Factorization (NMF), [online] https://qiita.com/takechanman1228/items/6d1f65f94f7aaa016377

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

[9] 小説家になろう, 2020 年 1 月 8 日アクセス, [online] https://syosetu.com

[10] gensim, 2020 年 1 月 8 日アクセス, 「models.word2vec-Word2vec embeddings」, [online] https://radimrehurek.com/gensim/models/word2vec