DL4US
Twitterのデータから
目的のカテゴリを持つデータを抽出する

Sekikawa

目次

1. 取り組んだ内容

ツイートのテキストから、カテゴリ分類を行う

「みずほ」を含むツイートから、以下の7カテゴリに分類分けする

- みずほ：人関連
- みずほ：口座売買関連
- みずほ：個人売買関連
- みずほ：投資関連
- みずほ：みずほ銀行関連 ★
- みずほ：アプリ関連 ★
- みずほ：その他、みずほ銀行に関係ないもの

2. 背景 – 1/4

Twitterのつぶやきはユーザのリアルな声が聞ける貴重なデータ

- 現状はTwitterの意見をキャッチアップ・反映が出来ていない
- 検索して、意味のあるTweetを探して、キャッチアップするのが手間

以下をキャッチアップするためにDLが活用できないか

- トレンドの把握
- 不満点の把握
- 良い点の把握
- ニーズの把握

2. 背景 – 2/4

考えていたこと

- Tweetのテキストからポジネガ分類を行う
  - ⇒ 不満点・良い点・ニーズを把握する

- Tweetのテキストからトピックに分類する
  - ⇒ トレンドの把握、トレンドの変化の把握
  - ⇒ アップデート直後にトピックがどれだけ変わったか

上記は取り組む前に詰まりました、、

2. 背景 – 3/4

詰まりポイント

- TwitterのAPIで取得できるのは特定の文字列が含まれるもののみ

- “みずほ”を含むTweetを取得すると、ノイズが多く含まれる
  - みずほ：人関連
  - みずほ：口座売買関連
  - みずほ：個人売買関連
  - みずほ：投資関連
  - みずほ：みずほ銀行関連 ★
  - みずほ：アプリ関連 ★
  - みずほ：その他、みずほ銀行に関係ないもの

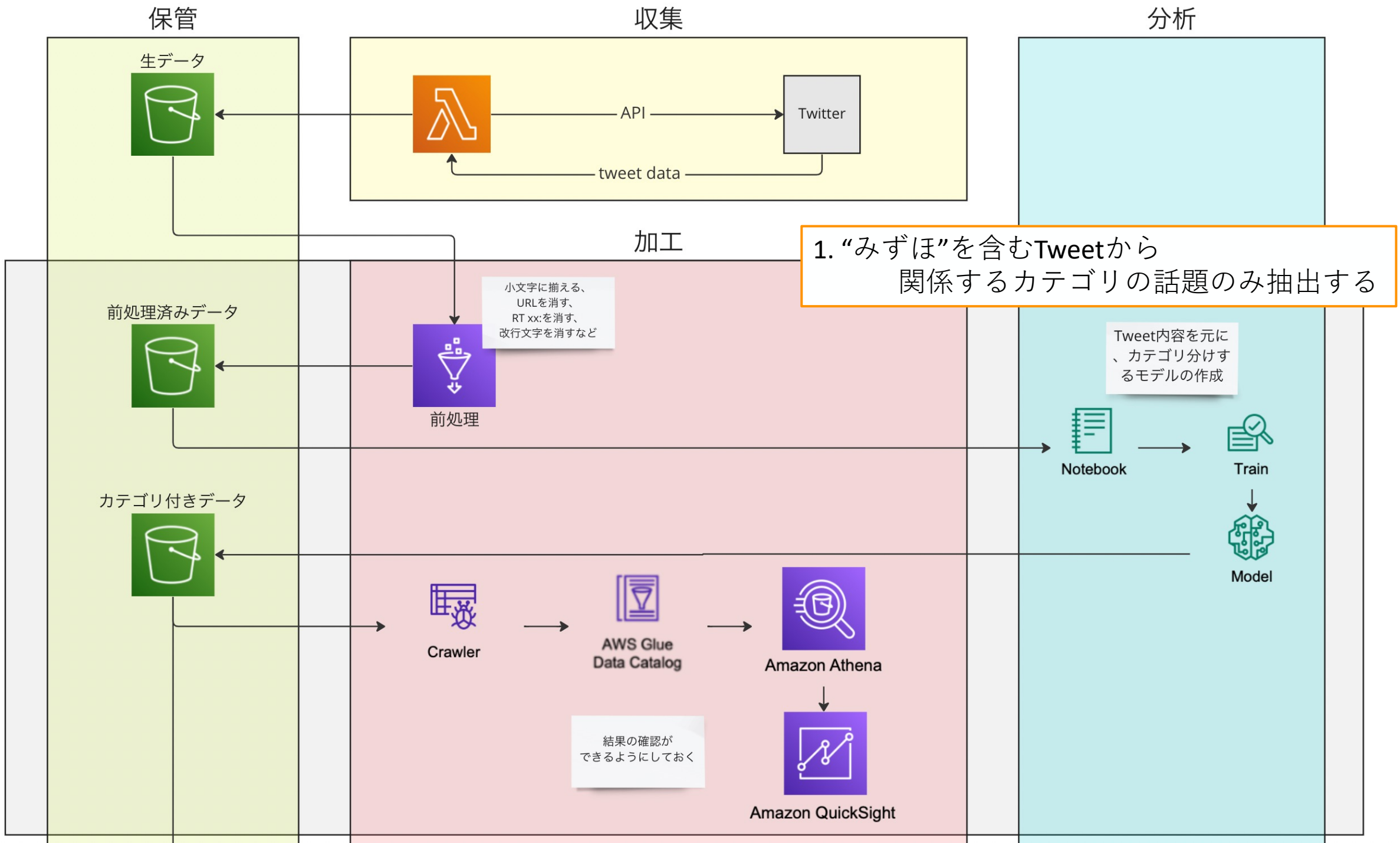
2. 背景 – 4/4

2段階で実現することを考える

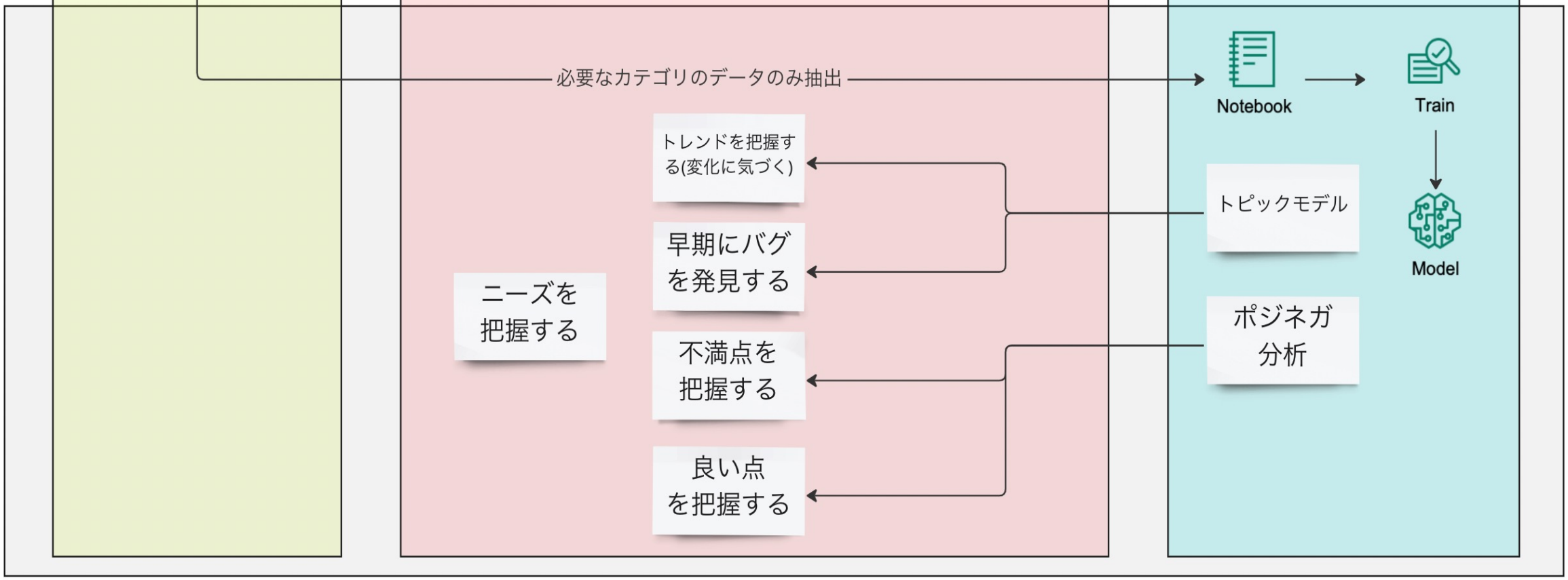
1. “みずほ”を含むTweetから関係するカテゴリの話題のみ抽出する
   - みずほ：アプリ関連
   - みずほ：銀行関連

今回はこっちのモデルを作成する

1. 必要なカテゴリに絞って、トピック・ポジネガ分類を行う
   - みずほ：アプリ関連のトピックモデル
   - みずほ：アプリ関連のポジネガ分類モデル


保管
収集
分析
生データ
API
Twitter
tweet data
加工
1. “みずほ”を含むTweetから
関係するカテゴリの話題のみ抽出する
小文字に揃える、
URLを消す、
RT xx:を消す、
改行文字を消すなど
前処理済みデータ
前処理
Tweet内容を元に
、カテゴリ分けす
るモデルの作成
Notebook
Train
Model
カテゴリ付きデータ
Crawler
AWS Glue
Data Catalog
Amazon Athena
結果の確認が
できるようにしておく
Amazon QuickSight

2. 必要なカテゴリに絞って、
トピック・ポジネガ分類を行う



3. 扱うデータ - 概要

Twitterの提供するAPIから取得したデータ

- 9/4〜10/22のTweetデータ
- データ量：51000件

- “みずほ”が含まれるTweetを15分おきに100件ずつ取得する
  - (同じTweet内容は削除)

- 7カテゴリに目視でラベル付(850件)
  - みずほ：人関連(189件)
  - みずほ：口座売買関連(57件)
  - みずほ：商品売買関連(30件)
  - みずほ：投資関連(44件)
  - みずほ：みずほ銀行関連(359件)
  - みずほ：アプリ関連(79件)
  - みずほ：その他、みずほ銀行に関係ないもの(92件)

3. 扱うデータ – ピックアップ

みずほ(人)カテゴリ

- みずほねぇさん だいすきほんとに優しい
- みずほさん仲間ですねっ 今週のしいたけ占い本当にしっくりきましたーっ
- ななことみずほさんが繋がったのおもろい仲良くしてねwwww

みずほ(口座売買)カテゴリ

- 困ってませんか　騙される前に　口座買取0年の実績　犯罪には絶対使いません　開設アプリから明日満額 窓口なら本日着金　開設もサポート　即日対応 三井0 中古相談　りそな0 新品満額　対応銀行増加 ゆうちょ 三菱 みずほ セブン 地方銀行 他の銀行も
- 口座買取りしてます　お気軽にご相談下さい　お待たせしません　dmお待ちしてます　三菱、みずほ ゆうちょ、三井、ui即金可能　詐欺など一切ございません

3. 扱うデータ – ピックアップ

みずほ(個人売買)

- 全てまとめて定価　送料　※振込先はufjもしくはみずほとなります
- 応相談　送金paypay、ゆうちょ銀行、みずほ銀行対応 0月頃発送されるものですので送り先のご提示は後日で大丈夫です

みずほ(投資関連)

- みずほリース絶賛含み損なう 下がったらコツコツs株で買い増ししてるけど駄目だずっと下がってるいい感じのところで入れるの祈ってまーす
- 利上げ局面の米新車販売 みずほリサーチ 金利上昇でも供給制約が解消に向かうことで来年は増加の見込み。加えて、これまでの繰延需要の発現や、供給回復による販売価格の低下などが需要を下支え

3. 扱うデータ – ピックアップ

多発している詐欺メール

みずほ(銀行関連)

- みずほ銀行の口座持ってないのに【みずほ銀行】お取引目的等のご確認のお願い　っていうメールが来た。

- 決済完了の音は「にゃん 」であってほしいなぁにゃんpay　『ヤマト運輸、独自スマホ決済「にゃんpay」みずほ銀行ハウスコインを利用』

10/6のアップデートで追加したもの

みずほ(アプリ関連)

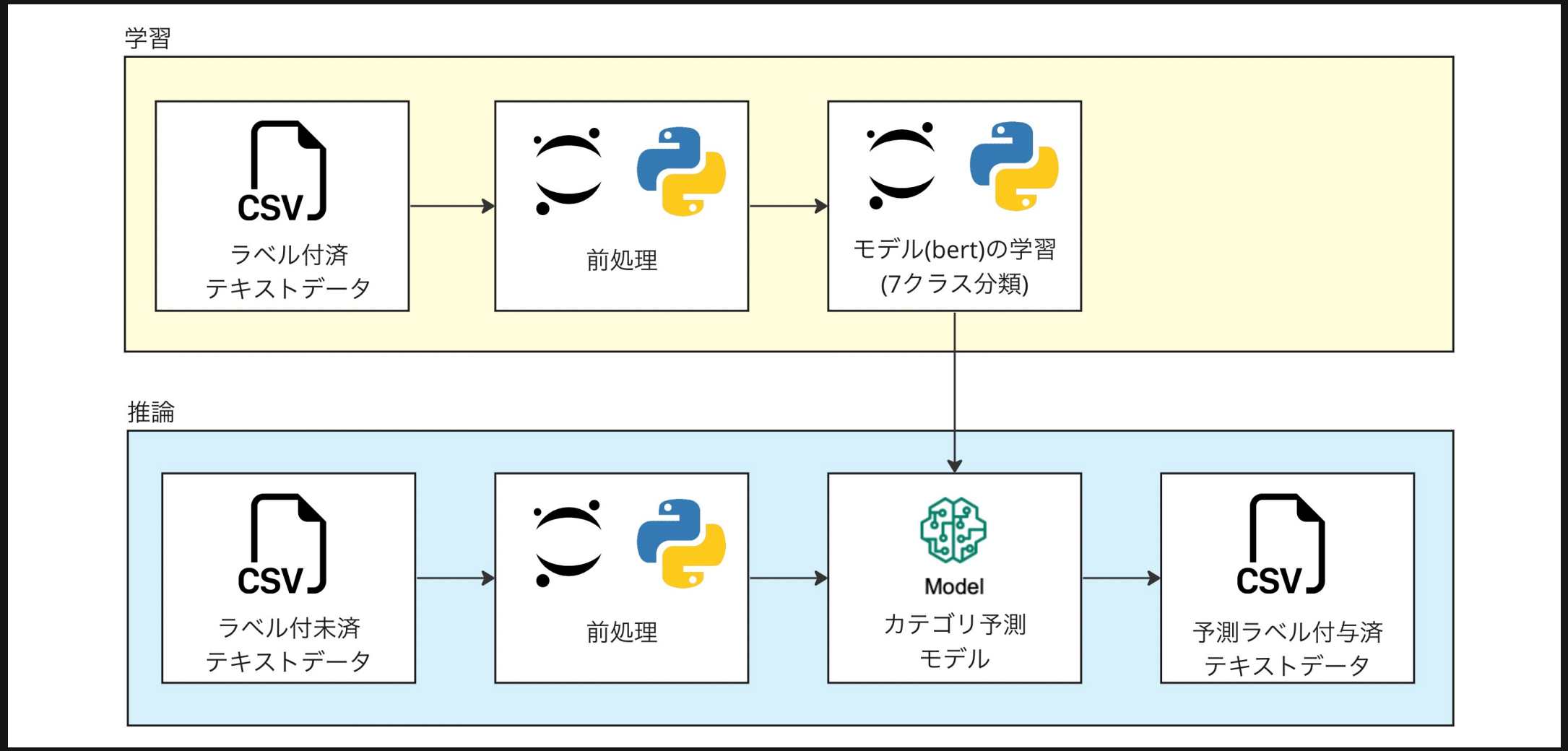
- みずほ銀行のアプリ、入出金明細ごとに残高確認できるようになったよ ってドヤ顔でお知らせしてきた…いや、待ち望んでたけどさ…

- みずほ銀行アプリの銀行振込やろうとするとメールで認証番号送ったから見てンゴwとかほざくんだけど、メール来て数字確認しようとするとアプリ閉じて入力全て取り消しになるんだが クソアプリ乙w

コールセンターに多く照会が入っているもの

実現方法 – 概要

BERTでTweetテキストを元に7クラス分類を行うモデルを作成する



実現方法 – データの前処理

取り除くもの

- 改行文字(¥n), URL, 絵文字
- “RT xxx :”

置換するもの

- 数字：全て0に
- 大文字：全て小文字に
- カタカナ：全角カタカナに統一
- 半角記号：半角スペースに置き換え
- 全角記号：半角スペースに置き換え

その他

- 前処理の結果、重複するTweetが発生したら取り除く

```python
def get_text_data(df):
    df["text"] = df["text"].astype(str)

    # 改行文字などを置き換える
    def remove_escape_str(t):
        return t.replace('\n',' ').replace('\r',' ').replace(" ", " ")

    # RT xxx :をから文字に置き換える
    def remove_RT(t):
        return re.sub("rt.+:", "", t)

    # URLを取り除く
    def remove_URL(t):
        return re.sub(r'https?://[\w/:%#\$&\?\(\)~\.=\+\-]+', '', t)

    # 絵文字を取り除く
    def remove_emoji(t):
        return demoji.replace(string=t, repl='')

    # 数字を全て0に置き換える
    def replace_num(t):
        tmp = re.sub(r'(\d)([,.])(\d+)', r'\1\3', t)
        return re.sub(r'\d+', '0', tmp)

    # 記号を置き換える
    def replace_symbol(t):
        # 半角記号の置換
        tmp = re.sub(r'[!-/:-@[-`{-~]', r' ', t)

        # 全角記号の置換 (ここでは0x25A0 - 0x266Fのブロックのみを除去)
        return re.sub(u'[■-♯]', ' ', tmp)

    # 大文字小文字を統一
    df["text"] = df["text"].map(lambda d: d.lower())

    # 文章のノーマライズ
    df["text"] = df["text"].map(lambda d: neologdn.normalize(d))

```

実現方法 – モデルの作成

BERTモデル

- “cl-tohoku/bert-base-Japanese”をファインチューニング
- 最終層から10層前までを学習する
- BERT部分と分類器部分で学習率を変える
  - BERT部分：低めの学習率
  - 分類機部分：高めの学習率

```
# BERTの最終層から10個前までのの層のみを学習する
for layer in model.bert.encoder.layer[:-10]:
    layer.trainable = False

# BERT部分と分類器で学習率を変える
optimizers = [
    tf.keras.optimizers.Adam(learning_rate=1e-5),
    tf.keras.optimizers.Adam(learning_rate=1e-4)
]
optimizers_and_layers = [
    (optimizers[0], model.bert.encoder.layer[:-1]),
    (optimizers[1], model.classifier)
]
optimizer = tfa.optimizers.MultiOptimizer(optimizers_and_layers)
```

実現方法 – モデルの評価

評価方法

- train：valid = 8：2 = 680：170
- 評価指標：Accuracy ⇒ 84%程度

Confusion Matrixも書いた方が良いはずだが時間切れ、、

```
Epoch 1/50
22/22 [==============================] - 43s 865ms/step - loss: 1.7169 - sparse_categorical_accuracy: 0.3668 - val_loss: 1.6456 - val_sparse_categorical_accuracy: 0.3488
Epoch 2/50
22/22 [==============================] - 13s 587ms/step - loss: 1.4208 - sparse_categorical_accuracy: 0.4833 - val_loss: 1.4345 - val_sparse_categorical_accuracy: 0.4186
Epoch 3/50
22/22 [==============================] - 13s 590ms/step - loss: 1.2273 - sparse_categorical_accuracy: 0.5852 - val_loss: 1.1499 - val_sparse_categorical_accuracy: 0.5988
Epoch 4/50
22/22 [==============================] - 13s 592ms/step - loss: 1.0188 - sparse_categorical_accuracy: 0.6579 - val_loss: 1.0057 - val_sparse_categorical_accuracy: 0.6221
Epoch 5/50
22/22 [==============================] - 13s 605ms/step - loss: 0.8892 - sparse_categorical_accuracy: 0.6783 - val_loss: 0.9047 - val_sparse_categorical_accuracy: 0.6512
Epoch 6/50
22/22 [==============================] - 13s 603ms/step - loss: 0.7559 - sparse_categorical_accuracy: 0.6987 - val_loss: 0.8182 - val_sparse_categorical_accuracy: 0.6628
Epoch 7/50
22/22 [==============================] - 13s 607ms/step - loss: 0.6314 - sparse_categorical_accuracy: 0.7671 - val_loss: 0.7156 - val_sparse_categorical_accuracy: 0.7791
Epoch 8/50
22/22 [==============================] - 13s 609ms/step - loss: 0.5287 - sparse_categorical_accuracy: 0.8457 - val_loss: 0.6503 - val_sparse_categorical_accuracy: 0.8023
Epoch 9/50
22/22 [==============================] - 13s 596ms/step - loss: 0.4443 - sparse_categorical_accuracy: 0.8836 - val_loss: 0.6731 - val_sparse_categorical_accuracy: 0.7965
Epoch 10/50
22/22 [==============================] - 13s 611ms/step - loss: 0.3795 - sparse_categorical_accuracy: 0.9083 - val_loss: 0.5988 - val_sparse_categorical_accuracy: 0.7965
Epoch 11/50
22/22 [==============================] - 13s 614ms/step - loss: 0.3230 - sparse_categorical_accuracy: 0.9287 - val_loss: 0.5896 - val_sparse_categorical_accuracy: 0.8081
Epoch 12/50
22/22 [==============================] - 13s 615ms/step - loss: 0.2660 - sparse_categorical_accuracy: 0.9534 - val_loss: 0.5771 - val_sparse_categorical_accuracy: 0.8198
Epoch 13/50
22/22 [==============================] - 13s 617ms/step - loss: 0.2328 - sparse_categorical_accuracy: 0.9607 - val_loss: 0.5341 - val_sparse_categorical_accuracy: 0.8430
Epoch 14/50
22/22 [==============================] - 13s 604ms/step - loss: 0.1905 - sparse_categorical_accuracy: 0.9723 - val_loss: 0.5681 - val_sparse_categorical_accuracy: 0.8140
Epoch 15/50
22/22 [==============================] - 13s 606ms/step - loss: 0.1530 - sparse_categorical_accuracy: 0.9840 - val_loss: 0.5433 - val_sparse_categorical_accuracy: 0.8256
Epoch 16/50
22/22 [==============================] - 13s 611ms/step - loss: 0.1262 - sparse_categorical_accuracy: 0.9825 - val_loss: 0.5457 - val_sparse_categorical_accuracy: 0.8256
Epoch 17/50
22/22 [==============================] - 13s 608ms/step - loss: 0.1171 - sparse_categorical_accuracy: 0.9811 - val_loss: 0.6015 - val_sparse_categorical_accuracy: 0.8081
Epoch 18/50
22/22 [==============================] - 14s 625ms/step - loss: 0.1103 - sparse_categorical_accuracy: 0.9840 - val_loss: 0.5328 - val_sparse_categorical_accuracy: 0.8372
Epoch 19/50
22/22 [==============================] - 13s 613ms/step - loss: 0.0853 - sparse_categorical_accuracy: 0.9927 - val_loss: 0.5493 - val_sparse_categorical_accuracy: 0.8430
```

実現方法 – モデルの改善

改善のためにやってみたこと

- 最終層からn層前までを学習する
  - ⇒ 大きな差異がなかったため10で固定

- 学習率/バッチサイズをいじる
  - ⇒ 大きな差異はなかった

- **ラベル付け済みデータを増やすのが最も効果的と思われる**
  - ⇒ 時間のある時にラベル付して学習データを増やす

今後のtodo

今回作成したカテゴリ分類モデルの改良について

- 学習データを増やす
- Confusion matrixを書いてみる
- Attentionの可視化をしてみる

みずほ(アプリ関連)のカテゴリについて

- トピックモデルを作る
- ポジネガ分類を行う

参考：ソースやラベルの予測結果など

GitHub

- https://github.com/Sekikawa318/documents/tree/main/twitter_analytics

- Google Colab上で動くはずです

ありがとうございました！