プログラミング演習 II (平成 30 年度)
課題-D

担当: 于, 真栄城

January 16, 2019

課題の提出について

ソースコード (プログラム) を「課題提出システム」で提出する. 提出前に, 自動チェックシステム, 実行結果チェックシステムを用いて問題がないことを確認すること. 木曜クラスは「rep13.exe」, 金曜クラスは「rep14.exe」を使用すること. 注意: ソースコードのファイル名を「kadai_d.rb」とする. 他のファイル名は受け付けない (未提出として扱う).

本課題では，紙媒体の提出物はない.

1 課題について

1.1 課題の概要

Twitter の投稿時間によりユーザ間の投稿パターンの類似度を計算する. まず，1 日を 24 の区間に分割する (0:00 から 0:59 まで，1:00 から 1:59 まで…23:00 から 23:59 まで). このときの分割数を N = 24 として，各ユーザの投稿パターンを 24 次元のベクトルで表現する.

ベクトルの各要素の値は Time-Weight(セクション 1.1.1 を参照する) という手法で計算する. 例えば，1:00 から 1:59 までの区間における全てのユーザの総投稿数が 10 回だと仮定する. ユーザ A が 1:00 から 1:59 までの区間に 4 回投稿した場合には, ユーザ A の投稿パターンを表すベクトルにおける 1:00 から 1:59 までの区間に対応する要素の値は $4 \times \frac{1}{10+0.2}$ と計算する.

投稿パターンを表すベクトルに基づいて、ユークリッド距離 (セクション 1.1.2 を参照する) を計算する. 2 つのベクトル間の距離が短いほど、2 つのユーザの投稿パターンは似ている.

1.1.1 Time-Weight

Time-Weight 法は、

$$Time - Weight(t) = c(t) \times \frac{1}{C(t) + 0.2} \tag{1}$$

- t：分割の区間
- $c(t)$：分割の区間 t に対応する時間中に解析しているユーザが投稿を投稿した回数
- $C(t)$：分割の区間 t に対応する時間中に解析している全てのユーザが投稿を投稿した回数

という 2 種類の統計情報をもとにして、ベクトルの各要素の値を計算する手法である.

1.1.2 ユークリッド距離

p $= (p_1, p_2, ..., p_n)$ および **q** $= (q_1, q_2, ..., q_n)$ が n-次元ユークリッド空間内の 2 点とすれば, **p** から **q** への, あるいは **q** から **p** への距離は

$$\sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + ... + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{2}$$

で与えられる. ユークリッド空間における点の位置は位置ベクトルで表されるから, さきの **p** および **q** は, 空間の原点を始点として終点がそれぞれの点であるような幾何ベクトルと見做すことができる. 詳細は Wikipedia[1] を参照する.

1.2 プログラムの概要

Ruby プログラムと同じディレクトリに「twitter_log_#.csv」という名前のファイルが複数あると仮定する．全てのファイルを読み込み，投稿データをユーザごとに集計し，投稿パターンのベクトルを作成する．各ベクトルの要素は同じ順序に並べる. 例えば、0 時台，1 時台，…，23 時台の順にすること．指定されたユーザと他のユーザとの投稿パターンの類似度を計算する.

1.3 入力ファイルのフォーマット

入力ファイル「twitter_log_#.csv」は，Twitter に投稿された投稿データを格納したテキストファイルである．ファイルの各行には以下のような形式で投稿データが保存されている.

```
2017–03–04,14:10:39,user00000001,“I am a football player”
```

ファイルには 1 行に 1 投稿の情報が格納されており,1 行は半角カンマによって以下の 4 つのフィールドに分けられている.

- 投稿日 (年月日)
- 投稿時間 (時分秒)
- ユーザ ID
- 投稿メッセージ

1.4 出力ファイル

「twitter_log_#.csv」という投稿データを読み込み，user00000001 というユーザの投稿パターンによって, 他のユーザを順位付けする.user00000001 との投稿パターンの類似度により, 以下のような降順で「order.txt」という名前のファイルに出力する. 類似度値を出力する時, 小数点第 5 位を「四捨五入」する.

[1] https://ja.wikipedia.org/wiki/ユークリッド距離

```
user00000002:3.5048
user00000004:2.4819
user00000003:1.0419
user00000005:0.3487
```

注意

- 本課題を行うためには，これまでの授業テキストで解説されていないメソッドを使う必要があるかもしれない．例えば，Ruby には配列やハッシュを辞書順にソートするメソッドが用意されている．必要に応じて，自分でメソッドを探し，使用すること．
- 変数名とメソッド名は他の人がプログラムを読んだときに意図がわかるような名前にすること．i, j のような 1 文字変数は使わない．例えば，繰り返し回数を保存する変数なら'index' や'count' など．
- コメントを付けること．新しく宣言する変数やまとまった処理，メソッドには必ずコメントを付ける．まとまった処理とは，条件分岐や繰り返しなどを指す．メソッド定義の部分では，(i) メソッドの機能，(ii) 引数の説明，(iii) 返り値の説明，の 3 項目に関するコメントを必ず記載すること．
- 一度にすべての機能を追加せず，処理をある程度の粒度に分割する．段階的にプログラムを作り，その都度正しく動作することを確認することを推奨する．