

Report

Vipin Reddy (24)

Q1.

Our application can collect the tweets according to the keyword specified (here the keyword is “food”). Then class “TwitterSentimentalAnalysis” will collect the tweets (count: 20) and class “SentimentAnalyzer” will do sentimental analysis for the collected 20 tweets. The analysis result will be printed out in the log.

Screenshots:

Analysis result

```
Adding annotator ssplit
Adding annotator parse
Adding annotator sentiment
TweetWithSentiment [line=Dealing with this chest pain, trying to eat fatty food, & watching Gotham with five of my mains., cssClass=sentiment : negative]
Adding annotator tokenize
Adding annotator ssplit
Adding annotator parse
Adding annotator sentiment
Adding annotator tokenize
Adding annotator ssplit
Adding annotator parse
Adding annotator sentiment
TweetWithSentiment [line=I'm happy asf because I just received food, cssClass=sentiment : positive]
TweetWithSentiment [line=The Food was as brilliant as it was involved, cssClass=sentiment : positive]
Adding annotator tokenize
Adding annotator ssplit
Adding annotator parse
Adding annotator sentiment
Adding annotator tokenize
Adding annotator ssplit
Adding annotator parse
Adding annotator sentiment
Adding annotator sentiment
TweetWithSentiment [line=Never had food from the strip club. Wonder if it's really good or not. They be advertising it like it is, cssClass=sentiment : negative]
null
Adding annotator tokenize
Adding annotator ssplit
Adding annotator parse
Adding annotator sentiment
Adding annotator tokenize
Adding annotator ssplit
Adding annotator parse
Adding annotator sentiment
null
TweetWithSentiment [line=Yeah no food for 24 hours really fucks you up, cssClass=sentiment : negative]
Adding annotator tokenize
Adding annotator ssplit
Adding annotator parse
Adding annotator sentiment
Mar 07, 2016 10:02:15 PM edu.stanford.nlp.process.PTBLexer next
WARNING: Untokenizable: ? (U+D83D, decimal: 55357)
```

Q2.

Step 1: Define 10 categories:

```
2::animal  
3::art  
4::book  
5::food  
6::movie  
7::music  
8::TV  
9::sport  
10::travel  
11::other|
```

Step 2: Collect tweets as training data to categorize the tweets into these 10 categories using keywords searching.

Step 3: Collect tweets again for recommendation training. In order to do rating, four items should be collected:

- 1) UserId. It's the tweet's userId which should be converted into integer.
- 2) Category. It is analyzed using feature extraction – TF-IDF by the training data collected in step 2.
- 3) Rating. Use sentiment analysis to give the rating for each tweet.
- 4) Timestamp. Tweet creation time which should be converted into integer.

These four items should be written into one file called “rating.txt”.

Step 4: Get the category mapping file called “category.txt”.

Step 5: Get the recommendation for one particular user. The recommendation is the categories that the user prefers.

Step 6: Send the results to the smartphone/smartwatch

Screenshots:

Example of getting category training data.

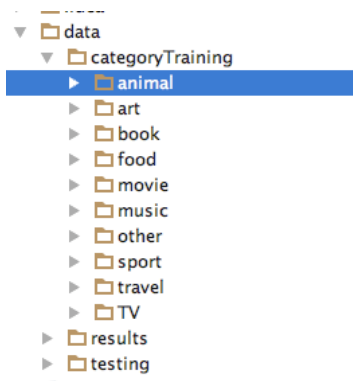
```
* Created by Ting on 3/9/16.
late more... (98F1)
public List<Status> GetCategoryTraining() {
    ConfigurationBuilder cb = new ConfigurationBuilder();
    cb.setDebugEnabled(true).setOAuthConsumerKey("R2v2WMKrF7UGipifRcMk0YjT1")
        .setOAuthConsumerSecret("InkVkJfUsJPQyA17GzGks9uzFSwUnRY9HqsR9m4vZ5Et3sW2d")
        .setOAuthAccessToken("3630687739-9y2qw6YK0MgeApmq09DK0uYosm2piadUy8aa96n")
        .setOAuthAccessTokenSecret("IBjoDz21BTBaXwnJ13jy2A0h0FaYzCYHmNRxCrhLLJong");
    TwitterFactory tf = new TwitterFactory(cb.build());
    Twitter twitter = tf.getInstance();
    Query query;
    query = new Query(keyword + " -filter:retweets -filter:links -filter:replies -filter:images");
    query.setCount(100);
    query.setLocale("en");
    query.setLang("en");
    try {
        QueryResult queryResult = twitter.search(query);
        return queryResult.getTweets();
    } catch (TwitterException e) {
        // ignore
        e.printStackTrace();
    }
    return Collections.emptyList();
}

public static void main(String[] args) throws IOException {
    GetCategoryTraining getCategoryTraining = new GetCategoryTraining();
    List<Status> statuses = getCategoryTraining.GetCategoryData("food");

    int i = 0;
    for (Status status : statuses) {
        if (status.getText() != null) {
            i++;
            File foodTextFile = new File("data/categoryTraining/food/" + i + ".txt");
            FileWriter fw = new FileWriter(foodTextFile);
            fw.write(status.getText());
            fw.close();
        }
    }
}
```

Finally 100 txt files for each category was collected.

File structure:



Category mapping:

```
2::animal
3::art
4::book
5::food
6::movie
7::music
8::TV
9::sport
10::travel
11::other|
```

Sentimental analysis and category analysis

```
public static void main(String[] args) throws IOException {
    TwitterSentimentalAnalysis twitterSentimentalAnalysis = new TwitterSentimentalAnalysis();
    List<Status> statuses = twitterSentimentalAnalysis.getTestingData();
    String a = "";
    int i = 0;
    for (Status status : statuses) {
        if (status.getText() != null) {
            i++;
            File newTextFile = new File("data/testing/1.txt");

            FileWriter fw = new FileWriter(newTextFile);
            fw.write(status.getText());
            fw.close();

            SentimentAnalyzer doAnalysis = new SentimentAnalyzer();

            int rate = doAnalysis.findSentiment(status.getText()).getRate();

            TwitterCategoryAnalysis twitterCategoryAnalysis = new TwitterCategoryAnalysis();
            int category = twitterCategoryAnalysis.CategoryAnalysis();

            int usrId = (int)((status.getId() >>> 32) ^ status.getId());
            int time = (int)((status.getCreatedAt().getTime() >>> 32) ^ status.getCreatedAt().getTime());
            a += usrId + "::" + Integer.toString(category) + "::" + Integer.toString(rate) + "::" + time + "\n";
            System.out.println(a);
        }
    }
}
```

Collect training data for recommendation according to above analyzes and write into file.

```
try
{
    String filename= "data/results/rating.txt";
    FileWriter fw2 = new FileWriter(filename,true); //the true will append the new data
    fw2.write(a);//appends the string to the file
    fw2.close();
}
catch(IOException ioe)
{
    System.err.println("IOException: " + ioe.getMessage());
}
```

Recommendation of user's category preference(partial code):

```
object MakeRecommendation {
    def main(args: Array[String]) {

        System.setProperty("hadoop.home.dir", "F:\\winutils")
        Logger.getLogger("org.apache.spark").setLevel(Level.WARN)
        Logger.getLogger("org.eclipse.jetty.server").setLevel(Level.OFF)

        if (args.length != 2) {
            println("Usage: /path/to/spark/bin/spark-submit --driver-memory 2g --class MovieLensALS " +
                "target/scala-*/movielens-als-ssembly-*.jar movieLensHomeDir personalRatingsFile")
            sys.exit(1)
        }

        // set up environment

        val conf = new SparkConf()
            .setAppName("CategoryALS")
            .set("spark.executor.memory", "2g").setMaster("local[*]")
        val sc = new SparkContext(conf)

        // load personal ratings

        val myRatings = loadRatings(args(1))
        val myRatingsRDD = sc.parallelize(myRatings, 1)

        // load ratings and category titles

        val categoryHomeDir = args(0)

        val ratings = sc.textFile(new File(categoryHomeDir, "rating.txt").toString).map { line =>
            val fields = line.split("::")
            // format: (timestamp % 10, Rating(userId, categoryId, rating))
            (fields(3).toLong % 10, Rating(fields(0).toInt, fields(1).toInt, fields(2).toDouble))
        }

        println(ratings)

        val categories = sc.textFile(new File(categoryHomeDir, "category.txt").toString).map { line =>
            val fields = line.split("::")
            // format: (categoryId, categoryName)
            (fields(0).toInt, fields(1))
        }.collect().toMap

        val numRatings = ratings.count()
        val numUsers = ratings.map(_._2.user).distinct().count()
        val numCategories = ratings.map(_._2.product).distinct().count()

        println("Got " + numRatings + " ratings from "
            + numUsers + " users on " + numCategories + " categories.")
    }
}
```

Result log

```
16/03/10 00:58:07 INFO JniLoader: already loaded netlib-native_system-osx-x86_64.jnilib
RMSE (validation) = 0.16005871649500103 for the model trained with rank = 8, lambda = 0.1, and numIter = 10.
RMSE (validation) = 0.18066115598146681 for the model trained with rank = 8, lambda = 0.1, and numIter = 20.
RMSE (validation) = 3.692744729379982 for the model trained with rank = 8, lambda = 10.0, and numIter = 10.
RMSE (validation) = 3.692744729379982 for the model trained with rank = 8, lambda = 10.0, and numIter = 20.
RMSE (validation) = 0.15739262495617246 for the model trained with rank = 12, lambda = 0.1, and numIter = 10.
RMSE (validation) = 0.180025657757478 for the model trained with rank = 12, lambda = 0.1, and numIter = 20.
RMSE (validation) = 3.692744729379982 for the model trained with rank = 12, lambda = 10.0, and numIter = 10.
RMSE (validation) = 3.692744729379982 for the model trained with rank = 12, lambda = 10.0, and numIter = 20.
The best model was trained with rank = 12 and lambda = 0.1, and numIter = 10, and its RMSE on the test set is 1.8075950816211317.
The best model improves the baseline by -61.31%.
Categories recommended for you:
1: art
2: travel
3: TV

Process finished with exit code 0
```

Results sent to smartphone:

