Software Requirements Specification

(MovieOracle System)

Revision History

Version	Date	Description	Author
1.0	Sep 10, 2010	first draft	Yao Yao
2.0	Oct 11, 2010	second draft	Yao Yao
3.0	Feb 12, 2011	third draft	Yao Yao
4.0	Mar 07, 2011	fourth draft	Yao Yao
5.0	Apr 20, 2011	fifth draft	Yao Yao
6.0	May 13, 2011	sixth draft	Yao Yao
7.0	Jun 07, 2011	seventh draft	Yao Yao
8.0	Jun 26, 2011	eighth draft	Yao Yao
9.0	July 22, 2011	ninth draft	Yao Yao

Table of Contents

1. Introduction

1.1 Purpose

All software requirements of the MovieOracle system are specified in this document. Its successor is the system detail design document.

1.2 Document Conventions

The first initial of each component name is in capital. The terms service and system are interchangeable in this document.

1.3 Business Opportunity

The system targets predicting Twitter users' evaluations on movies. It can be applied in marketing investigation, advertisement marketing and similar business applications. For example, with its help, a DVD rental company can find the twitter users (ids) who are predicted to be interested in a certain movie. Therefore, the company can do marketing activities to these potential users via tweet messages.

1.4 User-Level Goals

The users need a system to fulfill these goals:

1. continuously collect real-time tweets
2. classify tweets
3. predict twitter users' opinions
4. make recommendations on movies

2. Product Overview

2.1 Product Perspective

A recommender system predicts the attitude of a user towards an item. As to this project, it makes recommendations on movies to twitter users. First, the system receives movie names, then it retrieves tweets about the movie and finally it predicts users who will be interested in those movies.

2.2 Product Functions

The recommender system is given a group of movie names and then it collects tweets referring to such movies via the Twitter API. For each tweet, it finds the authors' friends and followers (these friends and followers are called "potential users") via the Twitter API (more introduction of the Twitter API can be found in Section 3 of the Detailed Design document). For potential users, tweets (on the same movies) published by their friends and followers are called “related tweets”. A potential user with too few related tweets is not processed.

For each tweet, a sentiment classifier [1] is used to determine its polarity - negative, positive, or unknown. For each potential user, the system calculates the percentages of three types of related tweets.

After the percentage calculation, MovieOracle uses a decision tree [2] to predict potential users’ evaluations for the movie. The percentages are used as attributes for the decision tree.

Also, the system outputs to a file all potential users who are predicted to hold positive opinions on a specific movie.

2.3 Design and Implementation Constraints

1. The database may affect system performance as data increases.
2. The accuracy of the polarity classifier may affect the final prediction accuracy.
3. Unrelated tweets may affect the prediction accuracy. The MovieOracle cannot guarantee that all tweets it collects are definitely related to movies. For example, a tweet referring “Harry Potter” may possibly talk about the book, not the movie.
4. Missing tweets may affect the prediction accuracy. The MovieOracle cannot guarantee that it collects all tweets talking about specific movies. For example, Twitter API does not allow collecting past tweets.
5. With current training data, the accuracy of the decision tree is around 0.6.

2.4 Assumptions and Dependencies

1. The training data set for the sentiment classifier can be replaced to acquire more accurate predictions.
2. The training data set for the decision tree can be replaced to acquire more accurate predictions.

3. System Features

The system is composed of six major components: System Service, Tweet Collection, Potential User Finder, Polarity Classifier, Decision Tree, and Database. Among these components, System Service receives its input data from the keyboard and invokes the other components to process data. Tweet Collection receives its input data via the Twitter API as well as the System Service and makes use of the database to record its results. Potential User Finder gets its input data both from the database and the Twitter API. It uses the database to record its output data. The other two, Polarity Classifier and Decision Tree, get their input data from the database and copy their output data to the database or to files. Figure 1 displays a use case diagram of the entire system.

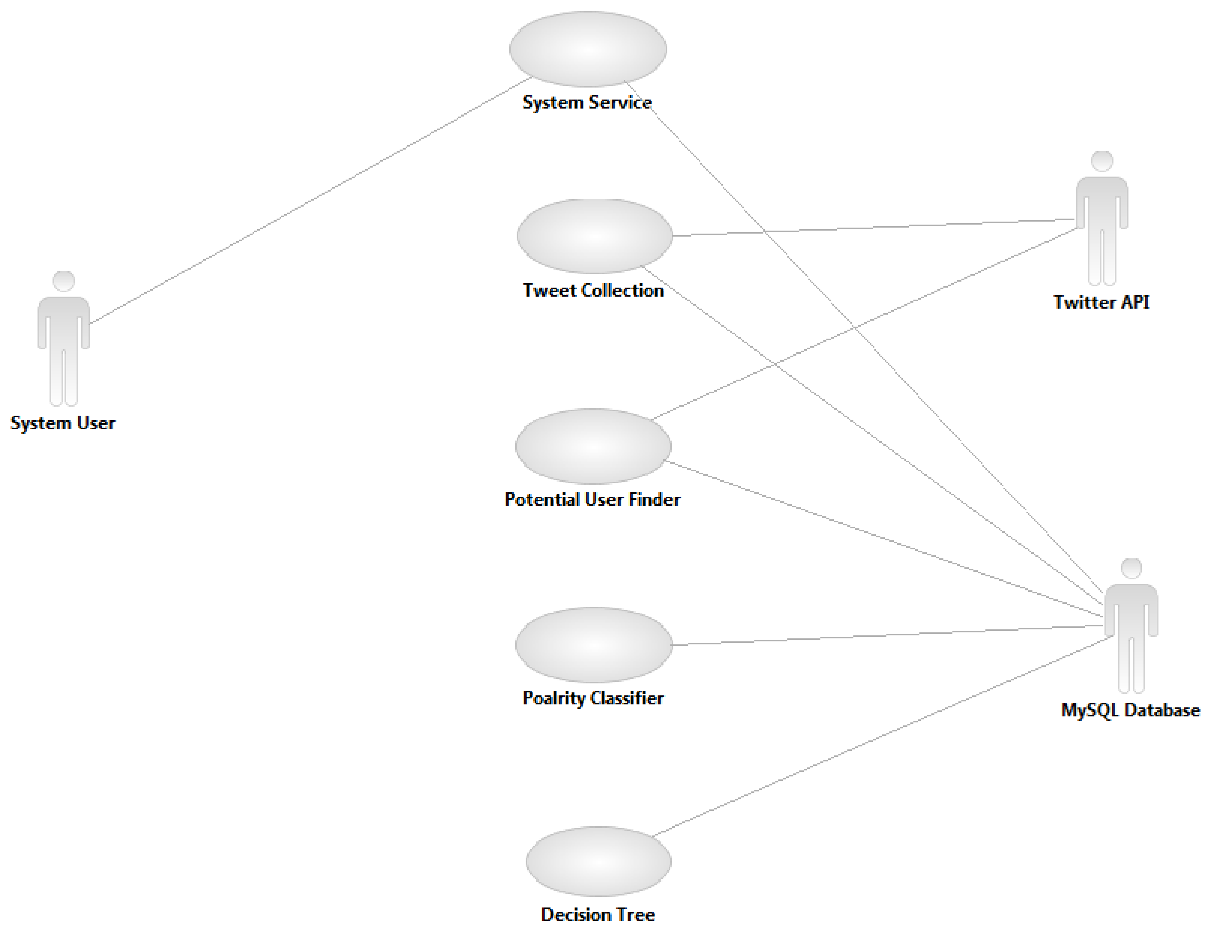

Figure 1: Use Case Diagram

3.1 System Service

3.1.1 Description

The System Service component invokes and coordinates other components. It also provides a command line console and other necessary services for the entire system.

3.1.2 Major Features

- invoke all of the other components
- provide a command-line user interface to control the following:
  (1) start/stop collecting tweets on one or a group of movie names
  (2) predict one or more twitter user's opinion on one or more movies
  (3) display all the movies being processed
  (4) stop the system
- provide a log service for the entire system

- provide a simplistic prediction method when the decision tree method is not applicable

The implementation can be found in Section 5.6 and 5.7 of the Detailed Design document.

3.2 Tweet Collection

3.2.1 Description

The Tweet Collection component receives movie names and then collects new published tweets containing the names of the movies. It also gathers tweet status id, tweet author id, and tweet author name. Information collected is stored in the database. For each movie, the collection task runs until a command stops it.

3.2.2 Major Features

- collect tweet information on indicated movies
- handle the problem of duplicate movie names when collect tweets on a movie
- handle the problem that the movie is not currently processed when stop collecting tweets on it

The implementation can be found in Section 5.2 of the Detailed Design document.

3.3 Polarity Classifier

3.3.1 Description

The Polarity Classifier component uses a sentiment classifier[1] to continually determine new tweets' polarities. It classifies tweets in three categories: positive, negative, and unknown. The checking time interval is determined by the user. The classification results are written to the database.

3.3.2 Major Features

- ignore positive or negative word(s) in the movie name that may impact classification
- periodically classify new tweets

The implementation can be found in Section 5.3 of the Detailed Design document.

3.4 Potential User Finder

3.4.1 Description

The Potential User Finder component continually checks the authors of new tweets. It acquires the friends and followers (called potential users) of these new-found authors. It further selects potential users with sufficient friends and followers with published tweets on the same movie. For these potential users, it invokes the Decision Tree service to predict their opinions.

3.4.2 Major Features

- ensure availability of the Twitter REST API
- acquire friends/followers (potential users) of new authors
- select potential users with a threshold specified in a property file
- generate data (potential users with their relative numbers of positive, negative, and unknown related tweets) for the decision tree
- invoke the Decision Tree service on these data

The implementation can be found in Section 5.4 of the Detailed Design document.

3.5 Decision Tree

3.5.1 Description

The Decision Tree component acquires the percentages of positive, negative, and unknown related tweets for a user. The decision tree predicts the potential users' opinions on a specified movie. Finally, it writes prediction results to the database.

3.5.2 Major Features

- use ID3 algorithm [3] to generate the decision tree
- copy the users holding positive opinions to a file

The implementation can be found in Section 5.5 of the Detailed Design document.

3.6 Database

3.6.1 Description

The Database component supports all of the data models of the MovieOracle system. The data models (tables) must be created before the system starts.

3.6.2 Major Features

- support data getting/setting operations of the other components
- relational database

4. External Interface Requirements

4.1 User Interfaces

A system user communicates with the system by command line interface. The basic operations are introduced in 3.1.2 Major Features (of the System Service component).

4.2 Software Interfaces

The support libraries include: mysql-connector-java-5.1.7-bin.jar, ci-bayes-1.0.4.jar (a third party classifier), twitter4j-2.1.12.jar (Twitter API), and javolution-5.5.1.jar.

More information about twitter4j-2.1.12.jar and ci-bayes-1.0.4.jar can be found in Section 3 and 4 respectively, of the Detailed Design document.

References

[1] Joseph Ottinger. CI-Bayes https://ci-bayes.dev.java.net/ Open Source
[2] Jiawei Han and Micheline Kamber. Data Mining: Concepts and Techniques, Second Edition (Chapter 6.3.1)
[3] Jiawei Han and Micheline Kamber. Data Mining: Concepts and Techniques, Second Edition (Chapter 6.3.2)