Group 6

MovieProphecy

Project Report

Abhishek Chakraborty
Zachary Chapman
Yogesh Bodhe
Aravinda Kumar Reddy Yempada

Department of Computer Science
University of North Carolina at Charlotte

Index

1. Introduction

User generated content is reshaping the way in which people interact over the internet. People share information, ideas and opinions on social media. The ‘sentiment analysis’ or ‘opinion mining’ of such user content could provide invaluable information for individuals as well as many organizations to grow their business.

We are particularly interested in analyzing data regarding recent and upcoming movies from the social network Twitter in order to generate their reviews. Existing movie surveys and review websites collect user and professional critic viewpoints or require users to submit one through forms. By harnessing user opinions directly from Twitter, we hope to eliminate extra steps in the aggregate survey field and calibrate user opinions in a more accurate and legitimate way.

2. Project Objectives

The primary objective of this project is to generate the reviews of the recent and upcoming movies by predicting their success based on the twitter data related to them. Any user can search a movie and see its review in terms of the percentage of success. A registered movie executive would get additional privileges to see the review using charts, timeline and actual analyzed data from twitter.

2.1. Scope of the Project

The scope of this project is:

1) To gather tweets related to the movie in question from Twitter.
2) To categorize the tweets as positive, negative or neutral.
3) To store the analyzed tweets in a persistent storage i.e. in a database.
4) To predict the success of the movie based on the ratio of positive tweets to negative ones.

3. Requirements Specification

3.1. Software and Hardware Resource Requirements

1) 1 GHz 32-bit (x86) or 64-bit (x64) processor
2) 512 MB of system memory
3) Hard drive with at least 1 GB of available space
4) Internet Information Services (IIS)
5) GlassFish Server 4.1
6) SQL Server 2012
7) Java (JDK/JRE) 6 or above
8) .NET framework 4.5
9) Internet access
10) Web Browser

3.2. Other Requirements

You can see the complete requirements specification document here. The document includes product description with its context, user characteristics, underlying assumptions, constraints and dependencies. It also contains detailed specification of functional, user interface, usability, performance, system interface/integration, security, data management and portability requirements.

4. Design

You can see the complete technical design document here. The document contains overall architecture of the system including class diagram, sequence diagram, activity diagram, state machine diagram and component diagram.

4.1. Evaluation of the Design

4.1.1. Architectural Evaluation

The system has 4 basic sub-modules:

1) Crawler
   We used Twitter4j API used to perform the task of fetching data from twitter. It uses OAuth (Open Authentication) protocol to authenticate the application registered by the twitter user before making actual requests to get the data. It is a very secure way which lets the user access the resources of twitter such as their servers and databases without actually sharing their details.

2) Classifier
   LingPipe classifier is used for sentimentally analyzing the tweets gathered by the crawler. The tweets are classified as positive, negative or neutral according to their sentiment (i.e. emotion). It uses powerful machine learning algorithms to carry out the polarity analysis of the tweets. Moreover, this tool is specially designed to perform sentiment analysis of movie related data.

3) Storage
   The system uses Microsoft SQL Server 2012 as the backend database management system. It provides us with the options of high availability and advanced security with powerful query engine.

4) User Interface (UI)
   The system has a simple but elegant user interface implemented using ASP.NET framework. A user can search a movie name using a text box or using an alphabetical list. It also displays a list of upcoming movie names so that the user can quickly glance through it to see if the desired movie name is present in it. This functionality uses the Rotten Tomatoes API to

get the movie list. A registered movie executive can see charts, timeline as well as some of the sentimentally analyzed tweets as a special privilege.

4.1.2. Structural Evaluation

The system uses Internet Information Services (IIS). It supports many applications protocols and authentication mechanisms. GlassFish is used as an application server. It is an open source server from Oracle with good performance, scalability and bidirectional client/server communication.

5. Methodology

The team members followed agile methodology for the implementation of the project. The project was completed in three 2-week sprint cycles. Our primary back end crawler and classifier were delivered after the first sprint. A basic UI with the ability to search any movie name was achieved after the second sprint. An improved UI featuring charts, timeline as well as dynamic movie name collection from Rotten Tomatoes was accomplished after the third sprint.

Early on we came up with our product backlog composed of user stories. None of us have experienced agile development in the past so it was difficult to decide how to come up with priorities. We originally attempted to come up with a fully represented system (both back and front end) during the first sprint. However, understanding and implementing the back end systems took up the full 2 weeks.

During the second sprint we found ourselves going back and forth on the best way to handle data storage. We finally settled on SQL Server 2012 as our best option for optimal data storage. It required us to preprocess our data into a SQL-friendly format for easy insertion and updates. Although we spent an unexpected amount of time revisiting the back end, we were able to get a basic UI and functionality established for our front end.

With the back end mostly settled, we were able to focus the majority of our effort in the third sprint towards improving the front end functionalities. We also considered hosting options both for our database and web application, but were unable to come up with a good option. The time spent researching hosting services would have taken away from implementing a working system, which was our primary goal. Because of this, everything currently runs locally on a single machine.

Agile methodology typically requires daily meetings for the team to go over status updates. Early on we attempted to hold meetings every Tuesday and Thursday as well as several online meetings on weekends. Varying schedules and responsibilities toward the end of the semester made it harder to meet as a group. We were, however, able to pull back together at the end to finalize our application and its documentation.

6. Implementation

The front end of the system is implemented using ASP.NET. The back end is implemented using Java. These systems are integrated using REST API as we faced challenges doing the same using web services. NetBeans, Eclipse and MS Visual Studio IDEs were used for coding purpose. MS SQL Server Management Studio was used for database schema design.

7. Testing

You can see the test case documents here. Unit and integration testing was performed on the system to make it as error free as possible. The unit testing was performed on all the sub-modules: searching, streaming, classifying and storage. The integration testing was done as end-to-end flows.

8. Future Scope

The current efficiency of the LingPipe classifier is about 75%. This efficiency can increased by implementing other machine learning techniques with higher efficiency to get more accurate polarity analysis of the twitter data.

The system is implemented as a web application. The system may be implemented as a mobile application in the future as there are more number of android and iOS application users.

Finally, any product needs to eventually provide income to its creators for it to be continued. Our original idea for commercialization was to put the various “executive” features behind a paywall but perhaps the most lucrative method of commercialization will be to host advertisements (these however are not mutually exclusive). Sites such as Rotten Tomatoes and IMDB feature full page advertisements for content related to movies and television.

9. References

Project inspiration: http://www.jscse.com/papers/vol3.no3/vol3.no3.46.pdf
Twitter Data Analytics: http://tweettracker.fulton.asu.edu/tda/TwitterDataAnalytics.pdf