University of Manchester
School of Computer Science
Project Report 2012

Text Mining Twitter for Software and User Perception

Author: Tariq Patel

Supervisor: Dr. Goran Nenadic

Abstract

Text Mining Twitter for Software
and User Perception

Author: Tariq Patel

The aim of the project is to investigate the performance of Gismos and to design and construct a super multi-functional Gismo.
The novel aspects of the new Gismo are described. The abstract should perhaps be about half a page long.
The results of testing, which show the abject failure of the Gismo, are presented.
In the conclusions proposals for rectifying the deficiences are outlined.

Supervisor: Dr. Goran Nenadic

Contents

1 Introduction **5**
- 1.1 Aim and Motivation . . . 5
- 1.2 Challenges . . . 5
- 1.3 Objectives . . . 6
  - 1.3.1 Collect and Filter Tweets by Keyword . . . 6
  - 1.3.2 Feature Extraction . . . 6
  - 1.3.3 Analyse Tweet Sentiment . . . 6
  - 1.3.4 Structure and Integrate Data . . . 7
  - 1.3.5 Visualise Data Through GUI . . . 7
  - 1.3.6 Evaluate System . . . 7
- 1.4 Report Structure . . . 7

2 Background **8**
- 2.1 Text Mining . . . 8
- 2.2 Sentiment Analysis . . . 8
- 2.3 Twitter Mining . . . 8

3 Design **9**
- 3.1 Methodology . . . 9
- 3.2 Use Cases . . . 9
  - 3.2.1 Use Case 1 . . . 9
  - 3.2.2 Use Case 2 . . . 10
- 3.3 General Architecture . . . 10
  - 3.3.1 Retrieving Tweets . . . 11
  - 3.3.2 Feature Extraction . . . 11
  - 3.3.3 Visualisation . . . 11

4 Implementation **12**
- 4.1 Tweet Retrieval . . . 12
  - 4.1.1 Twitter API . . . 12
- 4.2 Feature Extraction . . . 13
  - 4.2.1 Sentiment Analysis . . . 13
  - 4.2.2 URL Extraction . . . 14
  - 4.2.3 Tokenisation . . . 14
  - 4.2.4 Price Extraction . . . 14
  - 4.2.5 Part-of-speech (POS) Tagging . . . 14
  - 4.2.6 N-Gram Tagging . . . 14

4.2.7 Software Verification 15
4.3 Visualisation 15
4.3.1 Aggregation 15
4.3.2 Web Application 15

5 Testing and Results **16**
5.1 Testing 16
5.2 Results 16
5.2.1 GUI 16
5.2.2 Discussion 16

6 Evaluation **17**

7 Conclusions **18**

Bibliography **19**

List of Figures

2.1 Aspects of text mining 8

3.1 General architecture of the system 10
3.2 Design for tweet retrieval 11
3.3 Design for feature extraction 11

List of Tables

1.1 Challenges to be faced in this project 6
1.2 Complexity and priority of project objectives 6
1.3 Features to be extracted from tweets 7

Chapter 1

Introduction

The success of a piece of software is based largely upon user opinion. Gathering such information is conventionally done through means of surveying groups of users. However, in the days of social media, people generally express their opinions on popular social networks or microblogging sites such as Facebook and Twitter. This means it is now much easier for companies to receive feedback on products they have developed by monitoring these networks.

1.1 Aim and Motivation

Twitter has been at the core of many data mining projects in recent years and this is due to the sheer amount of data produced on a daily basis. Twitter users now post in excess of 340 million tweets every day[@tw12] and as such Twitter provides a massive corporus for opinion mining and sentiment analysis.

By text mining Twitter posts for software, users are able to discover new tools or programs they have not come across before, as well as see reviews by other users.

Thus, the aim of this project is to develop a system that text mines Twitter posts to find software or software development tools that have been mentioned by its users and to discover the general sentiment of users towards these softwares.

1.2 Challenges

There are many challenges facing Natural Language Processing(NLP)-oriented projects. Table 1.1 shows the key issues to be faced in achieving the main aim.

With the millions of tweets posted every day on Twitter, one can safely presume that many of these will have no relevance to software or any of the other desired information. As such it will be vital to ensure only the most relevant tweets are extracted from Twitter for analysis so as not to waste resources.

Another issue is the world-wide nature of the Internet and microblogging networks like Twitter. This means that several tweets will not be in English and for this reason it would be more difficult to extract features from these tweets. To counter this, it will be necessary to filter tweets not only based on key words but also on their language.

A major issue in NLP research is that of text message shorthand. In a formal document this problem becomes somewhat irrelevant due to proper usage of Standard English. However, when working with the Twitter platform, the service's 140 character limitation on tweets means

	Challenges
1	Finding relevant tweets
2	Non-English tweets
3	Text message shorthand

Table 1.1: Challenges to be faced in this project

	Task	Complexity	Priority
1	Collect and filter tweets by keyword	Simple	High
2	Feature extraction	Complex	High
3	Analyse tweet sentiment	Intermediate	Medium
4	Structure and integrate data	Intermediate	High
5	Visualise data through GUI	Intermediate	Low
6	Evaluate system	Complex	High

Table 1.2: Complexity and priority of project objectives

users are generally more likely to abbreviate their text and this allows for a lot of ambiguity in the context of each word, and variability in how users may say the same thing.

1.3 Objectives

In order to successfully complete a project of this magnitude, the task at hand must be split into smaller steps. These objectives are shown with their complexities and prorities in Table 1.2.

1.3.1 Collect and Filter Tweets by Keyword

Collecting tweets is a core task in this project as all work will be based on tweets stored in a database. Filtering through these is a relatively simple task in that it can be done using Twitter's APIs but there are some complexities in ensuring they are all relevant.

The main idea at this stage is to collect tweets from Twitter based on a set of key words and software names, games, programming languages, or company names stored in a dictionary in order to retrieve relevant, software-related tweets.

1.3.2 Feature Extraction

Feature extracting is the core functionality set out to be achieved in this project. Using rule-based text mining techniques, the aim of this task will be to retrieve up to nine features from every tweet, which are shown in Table 1.3.

1.3.3 Analyse Tweet Sentiment

Sentiment analysis is another of the more important tasks in this project. This is where tweets are analysed for subjectivity, i.e. whether the tweet is positive, negative or neutral, and this will be used to show the general user perception of each piece of software.

	Feature
1	Software name
2	Software version
3	Company or developer
4	Programming language
5	Operating system
6	Price
7	Relevant URLs
8	Tweet sentiment
9	Purpose of the tweet, e.g. marketing, review

Table 1.3: Features to be extracted from tweets

1.3.4 Structure and Integrate Data

The data needs to be structured and aggregated to be able to provide any meaningful output for the user. Without this step, the system is producing no useful information.

1.3.5 Visualise Data Through GUI

Visualising the data is a fairly low priority task in that the system first needs to gather the information. This project centres more around the core back-end development than user experience and as such only a simple user interface is needed in its initial stages.

1.3.6 Evaluate System

The final evaluation of the produced system will be key in determining the success of this project. The system will be evaluated on the basis of the accuracy of retrieved results, the relevance and novelty of information and general usability.

1.4 Report Structure

This report documents the implementation of a text mining system that is set out to achieve the previously stated goals. The remainder of this report has been split into 6 chapters. Chapter 2 details the general background of this project and previous work in the area. Chapter 3 goes into the design of the software implementation including use case analysis, the architecture of the system and the software engineering methodologies used. Chapter 4 details the process of implementing each stage of the project and goes into details of how specific aspects such as the Twitter API integration and feature extraction work. Chapter 5 details the testing methods, results and final outcomes of the project with any meaningful information gained. Chapter 6 provides the general evaluation of the finished project, also outlining the successes and failures of the task at hand. Finally, Chapter 7 details the author's conclusions of the project, with suggestions for further work and a summary of the report.

Chapter 2

Background

This chapter provides an overview of the text mining field along with previous work in the area and all necessary background information required to understand the major tasks involved in this project.

2.1 Text Mining

“Text mining is the process of extracting interesting information and knowledge from unstructured text"[HNP05].

2.2 Sentiment Analysis and Opinion Mining

2.3 Twitter Mining

There has been several previous works on text mining Twitter posts, however, the bulk of these have focussed solely on biomedicine and the financial sector.

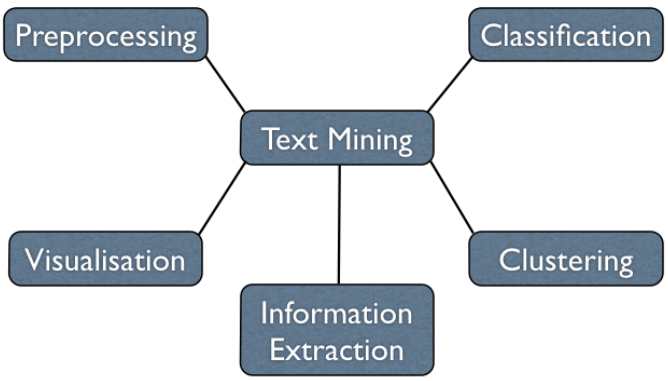

Figure 2.1: Aspects of text mining

Chapter 3

Design

This chapter details the overall design of the system to be developed in this project. The software engineering methodology to be used will first be discussed, along with use cases, requirements and the architecture of the system. This will be followed up with notes on the class and database design diagrams. The decisions made with regards to some design choices will also be discussed in more detail.

3.1 Methodology

The software engineering methodology used in developing an application can have many effects on its final outcome. The development of this system will be carried out using principles taken from continuous integration and agile methods such as feature-driven development. There is always a working code repository available for deployment, and all new features to be implemented are to be worked on in clones of said repository. Upon completion of these minor implementations, they are tested to ensure everything is working correctly and assuming there are no issues, the changes are merged into the base repository. This methodology assures developers that if any major issues arise due to recent changes, they will be able to discard all changes and restart if they feel debugging would be a longer process. This ultimately allows for a faster development cycle and provides rigorous testing throughout the implementation process. This development methodology also allows for frequently changing requirements which is to be expected in any development task.

3.2 Use Cases

There are two main use cases for the proposed system. The following use case definitions apply only to the first stage of the system, i.e. retrieving and storing posts from Twitter. In both cases the system requirements converge, and will be explored below.

3.2.1 UC1 - Streaming Twitter

The first use case for the system requires a tool capable of continuously monitoring public tweets and storing these in a database. These tweets should be filtered by language and relevance, that is, tweets should be related to software. These tweets need to be filtered by language

to counter any issues faced in the feature extraction stage due to the complexities involved in NLP. This use case will thus be referred to as streaming Twitter as that is its principle aim.

3.2.2 UC2 - Searching Twitter

The second use case for the proposed system requires the ability to search Twitter for tweets concerning user-specified key terms, and as such will henceforth be referred to as searching Twitter. These key terms should also be related to software. The returned tweets will also be filtered by language as in the streaming use case to counter the NLP complexities. An extra function required here should be to see which software tools have been mentioned most often, and these should be displayed to the users with the option to search again, or see the full analysis of these tools.

Upon fulfilling these core requirements, the systems should store these tweets in a database for work in the remaining stages. The first of these is extracting the previously stated features from these tweets. These features must again be stored separately from the initial tweet data, as some tweets may contain information regarding more than one piece of software. Finally, all of this information must be aggregated and shown to users in the form of charts displaying sentiment, and all relevant information found alongside it.

3.3 General Architecture

As previously explained, the system design follows a 3-stage approach, these being tweet retrieval, feature extraction and visualisation for users. These stages are shown in the general archicture model displayed in Figure 3.1. These will be further explored in Sections 3.3.1, 3.3.2 and 3.3.3 respectively.

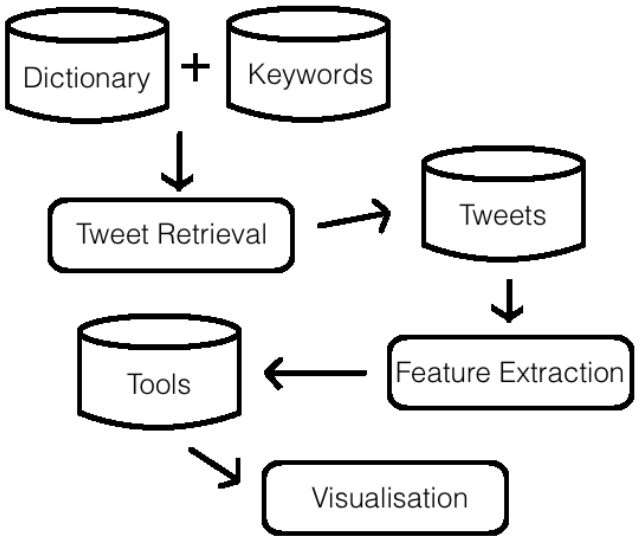

Figure 3.1: General architecture of the system

3.3.1 Retrieving Tweets

The first stage involves retrieving tweets from Twitter. The design for this stage follows the same concepts for each of the use cases defined. This can then be split further as seen in Figure 3.2. The program should retrieve a set of search terms from a dictionary along with some keywords that may be associated with software. These are to be used to form a request for tweets from Twitter. Twitter will respond with the corresponding tweets and data, which are to be checked for language, to ensure they are in English. The remaining set of English tweets are then to be further parsed to extract the required information for storing these tweets in the database.

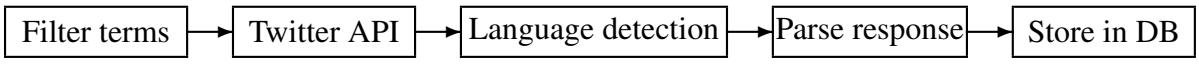

Figure 3.2: Design for tweet retrieval

3.3.2 Feature Extraction

The feature extraction stage will execute the task of extracting information from tweets. This will involve the stages described in 2.1 and its general design can be seen in Figure 3.3.

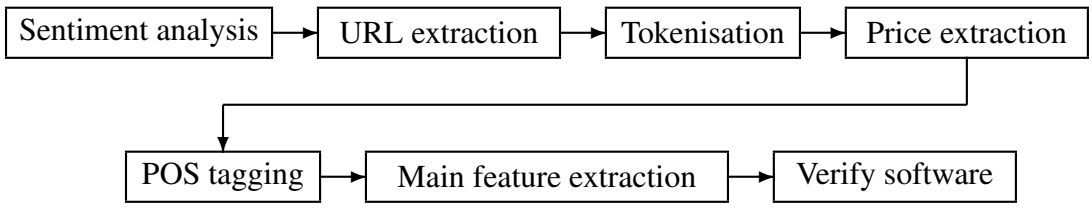

Figure 3.3: Design for feature extraction

3.3.3 Visualisation

Chapter 4

Implementation

This chapter outlines the main stages in implementating the designed system in code.

The system to be developed, as initially described in Figure 3.1, is fairly representative of a Question-Answering(QA) system. QAs are data mining systems which use Information Retrieval(IR) and Information Extraction(IE) techniques to answer user queries. As such, this project was implemented in 3 stages, each corresponding to these subsystems in QA systems. The first of these was retrieving tweets, the IR phase of this project. Upon successful retrieval, information, the required features in this case, must be extracted and finally, these results must be displayed to the user in a simple, straightforward fashion. These stages will be further explored as follows:

4.1 Tweet Retrieval

Without tweets, there is no work to be done, and so retrieving tweets can be regarded as the most important part of this project. The main objective of this stage is to retrieve as many relevant tweets from Twitter as possible. To do this, the system will interact with the set of public APIs Twitter provides.

4.1.1 Twitter API

Twitter provides three public APIs for developers to use. These are the REST, Search and Streaming APIs. The system is designed to use all of these to fulfill the requirements of each use case.

Streaming API

The Streaming API allows the system to fulfill the requirements of having a fully automated system that constantly monitors Twitter for software-related posts.

Search and REST APIs

The Search and REST APIs on the other hand allows user interaction in that users will be able to perform queries that will return up to 100 of the latest tweets in the last 7 days corresponding said queries.

4.2 Feature Extraction

Once tweets have been retrieved from Twitter and stored in the MySQL database, they are now available for feature extaction, which can be regarded as the core stage in implementing the system. This subsystem involves using NLP techniques to extract the information shown in Table 1.3 from each tweet. This subsystem is implemented in Python 2.7 due to the raw power it possesses and also due to the decision to use the Natural Language ToolKit(NLTK)[BLK09].

There are many steps involved in implementing the feature extraction. These are explored in order of execution.

4.2.1 Sentiment Analysis

Sentiment analysis was recognised as one of the key features to be extracted from the initial design stages. It has been implemented using the Twitter Sentiment Bulk Classification Service API. This was chosen ahead of others such as AlchemyAPI[Alc11] and the CLiPS Pattern modules. AlchemyAPI results were accurate, however with the massive number of tweets being streamed from the service it was not deemed feasible to continuously make calls to a web service to analyse them for sentiment, as the service only analyses individual tweets. As well as this, there is a limit of 10,000 tweets per day and with the large numbers of tweets posted on Twitter on the daily basis, this was also an issue.

While Pattern is an offline system, it states 72% accuracy for movie reviews[CLi12], while the Twitter Sentiment API is optimised for tweets and boasts 83% accuracy[GBH09]. As well as this, the bulk classification service allows mass analysis with requests consisting of up to 10,000 tweets.

The sentiment analysis of tweets is carried out before any of the other feature extraction work. As tweets have been retrieved and stored in the MySQL database, this part of the system selects 100 of the latest tweets, retrieving just the id and text, that have yet to be analysed and packs them into a JSON string object of the format:

```
{ "data": [ { "id": "1234", "text": "Google Chrome is awesome!" },
            { "id": "1235", "text": "Safari 5.0.2 is out now" },
            { "id": "1236", "text": "I really hate the new Firefox" } ] }
```

This JSON string is then posted to the Twitter Sentiment API where classifications into the positive, negative and neutral classes are carried out by a Maximum Entropy classifier trained with tweets containing emoticons. The internal specifics of a Maximum Entropy classifier, however, is not in the scope of this project.

Currently only 100 tweets are analysed at a time due to time constraints when users wish to run the program in real time. By using a small number, the data needing to be transferred is minimal and allows for a more interactive user experience.

Using the previous example, the data is returned by the server in the following format, with a polarity field added to each analysed tweet with values 0, 2 and 4 corresponding to negative, neutral and positive respectively.

```
{ "data": [
  { "id": "1234", "text": "Google Chrome is awesome!", "polarity": 4},
  { "id": "1235", "text": "Safari 5.0.2 is out now", "polarity": 2 },
  { "id": "1236", "text": "I really hate the new Firefox", "polarity": 0 }
] }
```

Upon receipt of this response, the JSON formatted string is parsed and the corresponding record for the tweet previously stored in the MySQL database is updated with new values for sentiment score and the actual sentiment, using polarity and its semantic meaning respectively.

4.2.2 URL Extraction

Before extracting context and semantics from tweets, any URLs mentioned are found and removed. Assuming the tweet is software-related, these URLs are quite likely to be links to the software, or further reviews. This task is done using NLTK's `regexp_tokenize` function with `http://[^ ]+` passed as the regular expression that finds URLs. If the tweet is later found not to contain any software, these URLs are discarded.

4.2.3 Tokenisation

After URLs have been extracted and removed from the source text, the tweet is tokenised to produce an array of all the terms in the tweet. The tokenisation process is also done using NLTK's `regexp_tokenize` function, passing it the regular expression - `\w+([.,]\w+)*|\S+`. This expression returned superior results to alternation tokenisation functions provided by NLTK, such as `wordpunct_tokenize` as it was capable of finding numbers and currencies without splitting them.

4.2.4 Price Extraction

Continuing on from this tokenisation of the original source text, the current subsystem attempts to find prices in the array of terms. This is done using Python's built-in regular expression module, `re`. A number of regular expressions are used to define patterns denoting numbers, currencies and quantifiers like 'hundred' and 'thousand'. As the form of prices vary, for example in the case of mobile apps you might find '£0.59', '59p' or even '59 pence', these combinations of tokens may be split across two tokens in the array returned from the tokenisation process. For this reason, it is necessary to iterate over all items in the list of tokens while remembering the previous one. This obviously means a less efficient system, however, it has produced the best results in such variable conditions.

4.2.5 Part-of-speech (POS) Tagging

The POS tagger used by this system is taken from the NLTK modules and uses the `pos_tag` function which takes a tokenised sentence as its only argument.

4.2.6 N-Gram Tagging

The n-gram tagging process consists of the core functions of the proposed system. Its purpose is to extract all the features that have yet to be extracted, that is, software names and versions, companies, programming languages, operating systems and the reason behind the tweet. It also attempts to find any prices that may previously have been missed, and also has the task of discovering software that is not already in the dictionary.

4.2.7 Software Verification

The feature extraction subsystem may discover new software, and as such needs to verify these are actually pieces of software and not something else. To do this the program utilises the Microsoft Bing API which returns web search queries. As the main tagging process checks the dictionary for matching software names, and the tweet retrieval engine uses both the dictionary and a set of keywords, there will be some pieces of software mentioned in the tweets that are not in the dictionary. As a result, these will be flagged as possible software names, and then queried on the Microsoft Bing search engine with the keywords "movie", "music", and "software game". These keywords were selected on the basis that the initial search key terms retrieved many tweets referring to music and films.

```
function bing_search(bing, term){
    music = bing.search(term, music)
    movie = bing.search(term, movie)
    software = bing.search(term, software game)

    if size(software) greater than size(movie) and size(music)
        if references to software in title and description
            return True
    return False
}
```

If the number of results for software associated with the searched term is greater than corresponding results for films and music, the results are checked for identifiers of software in their headings. Therefore if any of the results suggests the searched term is a piece of software, that is assumed true.

4.3 Visualisation/Graphical User Interface(GUI)

The final stage of the project is to aggregate and present the results to the user in a GUI. Aggregating the results is the process of bringing together all the different data sources for data on a single output entity such as a piece of software. This aggregated data can then be used easily by the GUI to display understandable information to the user. The GUI of choice is a web application as opposed to a desktop application, as it allows for a more centralised system that users can easily connect to. It is also a more scalable solution as updates would not need to be pushed out to all users.

4.3.1 Aggregation

4.3.2 Web Application

The web application available to users is a Python application running the CherryPy web framework.

Chapter 5

Testing and Results

This chapter details the testing methods use over the course of this project along with results and the final findings of the system.

5.1 Testing

5.2 Results

5.2.1 GUI

5.2.2 Discussion

Chapter 6

Evaluation

With implementation and analysis complete, one can now identify and evaluate the key successes and shortcomings of this project. These evaluations have been performed on the basis of the following questions:

- Does the system work?
- Is the information found novel and interesting?
- Is the system easy to use?

This process has been carried out by means of independent user evaluations by 5 users of the software.

Chapter 7

Conclusions

This chapter discusses the author's reflections on the success and conclusions of the project. Suggestions for further work are made and concludes with a summary of the report.

Bibliography

[Alc11] AlchemyAPI. Sentiment analysis. http://www.alchemyapi.com/api/sentiment/, October 21, 2011.

[BLK09] Steven Bird, Edward Loper, and Ewan Klein. *Natural Language Processing with Python*. O'Reilly Media Inc, 2009.

[CLi12] CLiPS. Pattern.en sentiment. http://www.clips.ua.ac.be/pages/pattern-en#sentiment, April 6, 2012.

[GBH09] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *Processing*, pages 1–6, 2009.

[HNP05] Andreas Hotho, Andreas Nürnberger, and Gerhard Paaß. A brief survey of text mining. *LDV Forum - GLDV Journal for Computational Linguistics and Language Technology*, 20(1):19–62, May 2005.

[@tw12] @twitter. Twitter turns six. http://blog.twitter.com/2012/03/twitter-turns-six.html, March 21, 2012.