SPOONS: NETFLIX OUTAGE DETECTION USING MICROTEXT CLASSIFICATION

A Thesis

Presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Eriq Augustine

March 2012

COMMITTEE MEMBERSHIP

TITLE:	SPOONS: Netflix Outage Detection Using Microtext Classification
AUTHOR:	Eriq Augustine
DATE SUBMITTED:	March 2012
COMMITTEE CHAIR:	Alex Dekhtyar, Ph.D.
COMMITTEE MEMBER:	Clint Staley, Ph.D.
COMMITTEE MEMBER:	Franz Kurfess, Ph.D.
COMMITTEE MEMBER:	Foaad Khosmood, Ph.D.

Abstract

SPOONS: Netflix Outage Detection Using Microtext Classification

Eriq Augustine

Every week there are over a billion new posts to Twitter services and many of those messages contain feedback to companies about their services. One company that has recognized this unused source of information is Netflix. That is why Netflix initiated the development of a system that will let them respond to the millions of Twitter and Netflix users that are acting as sensors and reporting all types of user visible outages. This system will enhance the feedback loop between Netflix and its customers by increasing the amount of customer feedback that is being received by Netflix and reducing the time it takes for Netflix to receive the reports and respond to them.

The goal of the SPOONS (Swift Perceptions of Online Negative Situations) system is to use Twitter posts to determine when Netflix users are reporting a problem with any of the Netflix services. This work covers the architecture SPOONS system and framework as well as outage detection using tweet classification.

Acknowledgements

Thanks Alex, thanks Netflix (Kevin). Negative thanks to Farscape.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 General Problem: Swift Perception Of Online Negative Situations

Twitter is an immensely popular micro-blogging service. According to Twitter, approximately one billion micro-posts, *tweets*, were being poster per week as of March 14th 2011[12]. Because of the low time and effort cost of tweeting, only a few seconds from a smart phone, Users of Twitter post tweets about almost every aspect of their daily lives. Because of this large stream of information, Twitter makes an excellent source of information for data miners. Already, researchers have been using Twitter to attempt to track and model disease outbreaks[2], earthquakes[8], and the stock market[7].

Netflix saw the power in this data source as a potential for detecting service outages that is orthogonal to their current outage detection methods. Currently, Netflix utilizes four different methods for detecting outages:

Internal Monitoring Systems Like any sizable service providing company, Netflix utilizes many different internal monitoring systems to detect service outages. However, there are some class of problems that are difficult to solve with internal monitoring. These problems include broken encodes of video files or a problem on a third-party delivery platform such as Roku or AppleTV. These problems are obvious to an end user, but very difficult to detect internally. In addition, the internal monitoring systems share the same infrastructure as the service providing system. Therefore, a problem in the infrastructre can cause both systems to go down at the same time.

External Monitoring Systems Netflix contracts with external services that can periodically probe its systems to try and detect problems. However, this model too has problems. There are many problems that cannot be seen from an external probe. Also, if this system probes too often then it is taking compute time away from the servers that are trying to deliver content to end users.

Customer Service Calls to customer service are a very straight-forward way to detect outages. Unfortunatley, this method is very slow and inconsistent. It takes a lot of frustration to get a user to lookup a phone number and complain.

Manual Twitter Observation Manual observation shows that there is usually a response on Twitter when Netflix suffers a service outage. Image 1.1 shows some tweets that occured during a disruption of Netflix's serice to the Nintendo Wii. However without any infrastructure, Twitter observation is slow and inconsistent. It is also very time consuming to have someone constantly watching Twitter for signs of an outage.

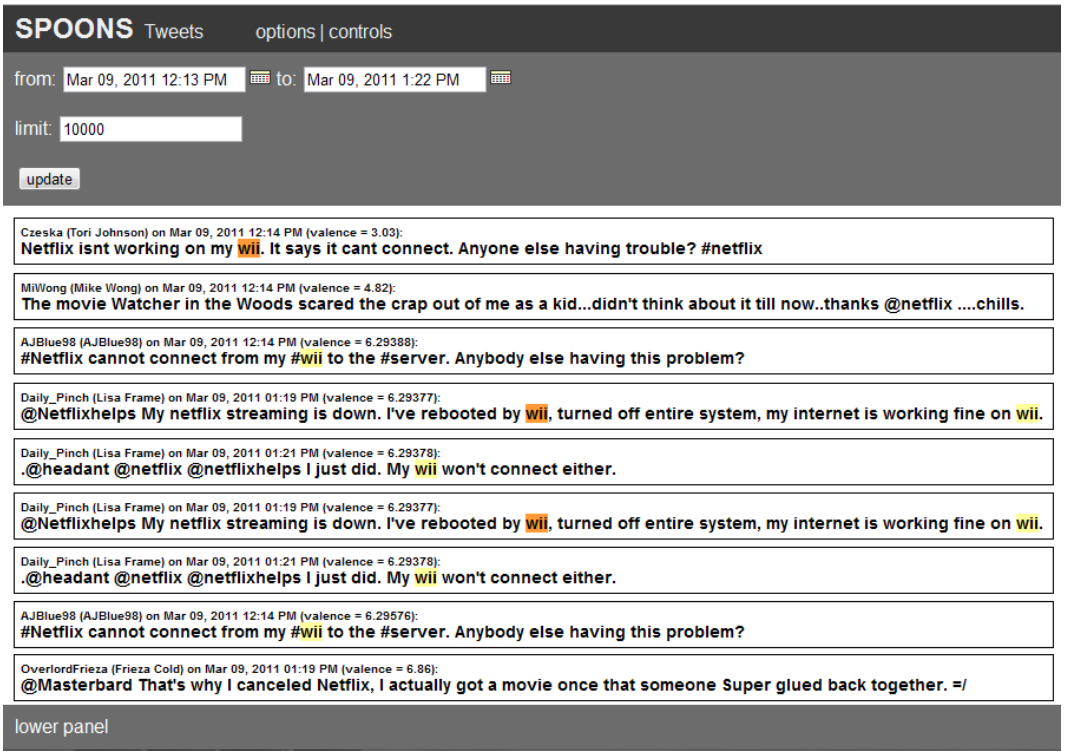

Figure 1.1: Tweets posted on March 9, 2011 during a disruption of Netflix streaming to the Nintendo Wii console.

Given all these deficiencies Netflix wanted a monitoring system that is seperate from their infrastructure, fast, and does not require any human intervention[9].

1.2 Solution Overview

SPOONS (Swift Perception Of Online Negative Situations) is a system that is designed to use tweets to detect outages in Netflix systems. The system supports a wide variety of detection methods that use some combination of time series analysis, classification, natural language processing, sentiment analysis, and filtering.

Image 1.2 shows how the SPOONS system can be divided into 3 main parts:

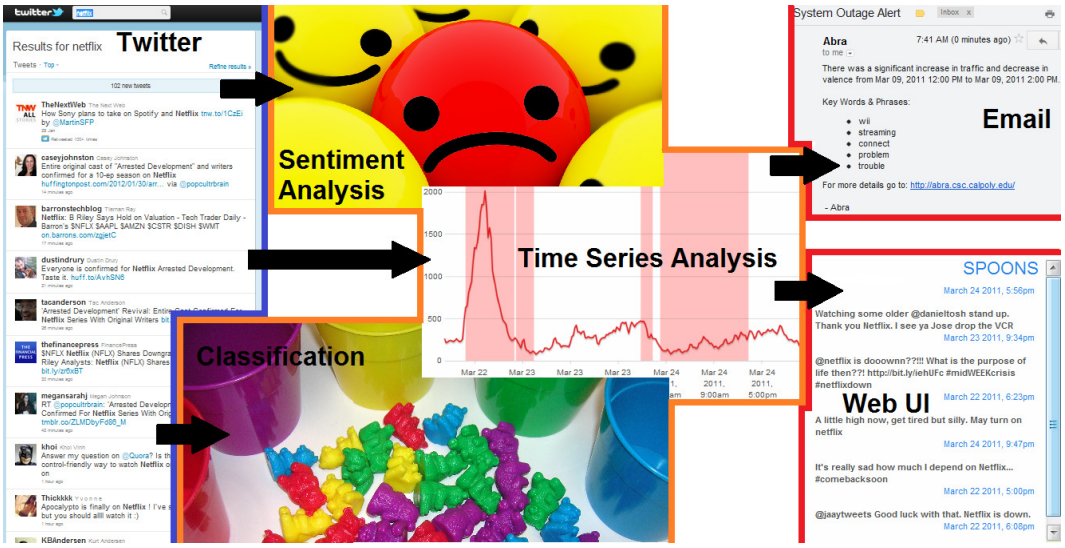

Figure 1.2: This system concept diagram shows the general flow of processing done in the SPOONS system.

input; analysis methods; and output. The inputs are tweets gathered from Twitter. Then the analysis methods use a combination of sentiment estimation, classification, and traffic volume analysis to detect when an outage is occurring. The outputs of the system are: email alerts to Netflix engineers; and a web UI that displays information about the outage.

1.3 Ethics of Twitter Observation

The work in this project uses content that people post on Twitter without their knowledge. This monitoring system isn't being announced to the public because wide spread knowledge of it would increase the likelyhood of a malicious attack. This practice may lead to concerns about the level of privacy or ownership being provided to Twitter users regarding the content they post through the Twitter services. The goal of this section is to address these concerns by proving more information about the Twitter services and how the SPOONS system and

this work uses the tweets.

1.3.1 Twitter Terms of Service

According to Twitter Terms of Service[13] agreement, that everyone accepts automatically by accessing or using Twitter services:

"You retain your rights to any Content you submit, post or display on or through the Services. By submitting, posting or displaying Content on or through the Services, you grant us a worldwide, non-exclusive, royalty-free license (with the right to sublicense) to use, copy, reproduce, process, adapt, modify, publish, transmit, display and distribute such Content in any and all media or distribution methods (now known or later developed)."

"This license is you authorizing us to make your Tweets available to the rest of the world and to let others do the same."

"You agree that this license includes the right for Twitter to make such Content available to other companies, organizations or individuals who partner with Twitter for the syndication, broadcast, distribution or publication of such Content on other media and services, subject to our terms and conditions for such Content use."

"We encourage and permit broad re-use of Content. The Twitter API exists to enable this."

"Such additional uses by Twitter, or other companies, organizations or individuals who partner with Twitter, may be made with no compensation paid to you with respect to the Content that you submit, post, transmit or otherwise make available through the Services."

To summarize, while Twitter users do own the content they post, by posting it through a Twitter service, they give Twitter and its partners rights to reuse it without compensation. As a user of the Twitter API, the SPOONS research group has become a partner of Twitter. So the analysis of tweets, extraction of tweet metadata, and aggregate use of that data is well within the rights of a partner of Twitter as defined by the Twitter Terms of Service.

1.4 SPOONS Requirements

Netflix has provided the following set of key requirements to be met by the SPOONS system:

Structural Independence The outage detection system shall be structurally independent of both the software and the hardware infrastructure used by Netflix. It shall rely only on information that is publicly available and free for use. This ensures that the outage detection system stays up even when any or all Netflix servers are experiencing downtime.

Use of Amazon Web Services Netflix is one of the largest customers of Amazon.com's cloud computing service, Amazon Web Services (AWS). AWS allows users to create new cloud machines (instances) in many regions throughout the world. The outage detection system shall be deployed on one or more AWS servers that are operationally independent of other AWS servers used by Netflix. Using a cloud solution allows the outage detection and alert system to be deployable on a global scale.

Real-Time Netflix's streaming services run in real-time and any downtime has an immediate impact on customers. To minimize that impact, the outage detection system shall notify Netflix of detected outages as soon as possible.

Precise Outage Detection The number of non-outage situations that raise an alert shall be minimized. While a small number of false positives detected in real-time may be acceptable, the outage detection system shall detect outages and generate alerts with as high precision as possible.

Comprehensive Outage Detection Not all Netflix outages will generate a signal on Twitter. Those that don't may be allowed to go unnoticed by the outage detection system (as the system will have no basis for detecting them), but any outage that causes a signal on Twitter shall be detected.

User-Friendly Online UI The outage detection and alert system shall have an easy-to-use, informative, online UI which shall provide Netflix employees with real-time information and historic data about the state of Netflix according to Twitter. The information provided shall include:

- times of outages;
- times of other anomalous events;
- current and recent Netflix-related Twitter traffic trends;
- and samples of Netflix-related tweets.

Chapter 2

Background & Related Work

2.1 Text Stream Analysis

Text Stream Analysis [1][3][6].

2.2 Classification

Classification [11]

2.2.1 Microtext Classification

Microtext Classification [5]

2.2.2 WEKA

SPOONS utilizes several classifiers provided in the WEKA machine learning package. WEKA is an open source package

Figure 2.1:

written under the GNU General Public License[4].

2.3 Twitter API

All of the data data that SPOONS uses is obtained in real time using the Twitter Search REST API[14].

2.3.1 Rate Limiting

Twitter imposes a limit on the number of queries to the Search API. However, Twitter does not publish the official limit. However, my experiments suggests that SPOONS can query the API for all new Tweets once every two minutes without suffering from rate limiting.

2.3.2 Pagination

Twitter paginates the results from its search api. The maximum results you can get per page is 100, and each query can return at most 15 pages. Therefore when there are more than 1500 tweets generated per minute, SPOONS must do multiple seach queries.

2.3.3 Query Anatomy

One of our typical queries looks like:
http://search.twitter.com/search.**json**?**q**=⟨query⟩&**rpp**=100&**result_type**=recent&**since_id**=⟨tw
id⟩&**max_id**=⟨tweet id⟩

json Twitter can supply the result data in either ATOM or JSON format. Testing with both have shown that the ATOM results are less consistent and provide less data. Because of the more accutate information returned from the JSON api, we are able to write more efficient queries. Using the ATOM api, we could query Twitter only once every five minutes; as opposed to every two minutes with the JSON api.

q The search query. Twitter supports some advanced search features such as conjunction and negation.

rpp "Results Per Page". Twitter paginates the responses from the Search API. SPOONS always uses the maximum pagination value to decrease the number of requests per hour and lessen the change of being rate limited.

result_type Twitter allows users to get results orderd search by either relevance or time. Since we want to gather all tweets about our query, we choose to get the results ordered by time. In addition, the "since_id" and "max_id" parameters do not work when results are sorted by relevance.

since_id The id of the oldest tweet that should be returned. This is not a hard limit, but provides a nice starting point.

max_id The id of the most recent tweet that should be returned. It may seem counter-intuitive to provide a cap on the most recent tweet, when one wants to query for all of the most recent tweets. However when a query spans across more than 15 pages, it will need to be broken into a new query restarting at the first page. In this situation, not providing an upper limit will include new tweets

```
{
   completed_in: 0.013,
   max_id: 257915466582990850,
   max_id_str: "257915466582990848",
   page: 1,
   query: "netflix+eriq",
   refresh_url: "?since_id=257915466582990848&q=netflix%20eriq",
 - results: [
     - {
          created_at: "Mon, 15 Oct 2012 18:46:53 +0000",
          from_user: "eriqaugustine",
          from_user_id: 882882438,
          from_user_id_str: "882882438",
          from_user_name: "Eriq Augustine",
          geo: null,
          id: 257915466582990850,
          id_str: "257915466582990848",
          iso_language_code: "da",
        - metadata: {
              result_type: "recent"
          },
          profile_image_url: "http://a0.twimg.com/sticky/default_profile_images/default_profile_4_normal.png",
          profile_image_url_https: "https://si0.twimg.com/sticky/default_profile_images/default_profile_4_normal.png",
          source: "&lt;a href=&quot;http://twitter.com/&quot;&gt;web&lt;/a&gt;",
          text: "eriq love netflix",
          to_user: null,
          to_user_id: 0,
          to_user_id_str: "0",
          to_user_name: null
       }
   ],
   results_per_page: 5,
   since_id: 0,
   since_id_str: "0"
}
```


Figure 2.2: A JSON result from the Twitter Search API

outside of the original search scope. This can result in tweets are forever lost to us.

2.3.4 Result Anatomy

Image 2.2 shows the result from the query "eriq netflix". Notice that some fields, like the "geo" field, can be null. Also note that the api incorrectly guessed my language as Danish.

Chapter 3

SPOONS Architecture

3.1 Architecture Overview

There are multiple levels of architecture within SPOONS that need to be discused. There is the Framework Architecture (Image 3.1) that describes the relations between the different pieces of the framework; the Server Architecture (Image 3.4) that describes the layout of the different servers involved in the SPOONS system; and the Distribution Model which describes how tasks are distributed between the different servers.

3.1.1 Claims

As part of my thesis, I lay claim to the design and implementation of the SPOONS system, framework, server architecture, distribution model, and database schema. However, I do not claim the UI and API. Those were the product of Matt Tognetti.

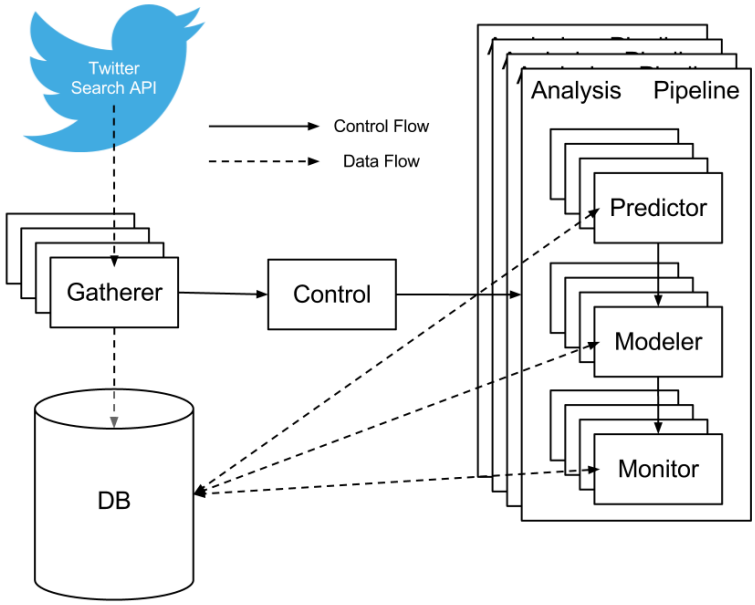

Figure 3.1: The flow of control and data through the SPOONS framework system.

3.1.2 Backend

The "backend" consists of all parts of SPOONS that are not part of the UI or API. This includes code that runs on multiple servers as well as the database.

3.1.3 UI and API

SPOONS includs a web UI that allows users to quickly and easily view the data produced by SPOONS. The UI is written primarily in PHP and Javascript and is seperated from the SPOONS backend. The only shared resource between the UI and backend is the database. The UI will automatically pull data from tables that have a specified schema and present them to the user.

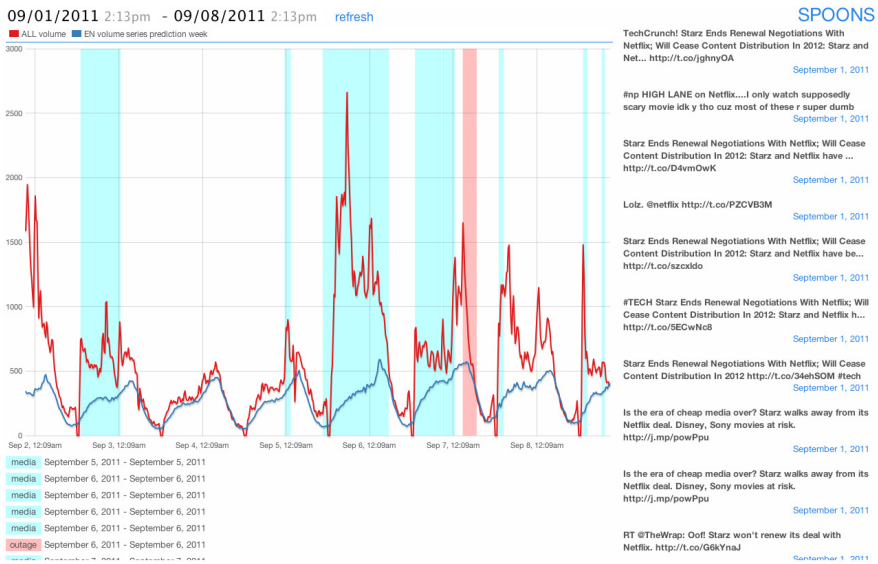

Figure 3.2: The web UI for SPOONS.

3.2 Framework Architecture

This section describes the architecture of the SPOONS framework. The SPOONS framework includes all pieces of SPOONS that takes the data from gathering all the way through to final analysis.

3.2.1 Gatherers

The data begins its journey in the Gatherers. The Gatherers run periodically (for Twitter, every two minutes). Gatherers are asynchronous and not dependent on any other part of the framework. There may be multiple different Gatherers running on the same machine. Gatherers are abstracted to be able to gather data from any source. Once the Gatherers get their data, they place the data in the database and notify the Control that there is new data available to the system.

Twitter Holes

It is worth noting that sometimes the Twitter Search API fails to return any data. We have not discovered the cause of this, but Twitter does not report any errors. For unspecified amounts of time the Twitter API will just report zero new tweets. We call these dead zones "holes". We have found that a query from a different IP usaully does not experience the same hole. To counteract holes, we run Gatherers on multiple servers and resolve uniques upon insertion into the database.

3.2.2 Processors

Processors are responsible for processing or transforming data before it goes into the analysis pipelines. Processing the data could be any task. Some of our most notable Processors are ones that classifies tweets into one of the nine tweet categories discussed in Section 4.5.

Unlike most parts of the analysis pipeline, Processors are a shared resource. That is, multiple analysis pipelines invokle the same processors. However, it does not make sense to restart the processing once it is started, or to start another instance of the same Processor. Processors have a finite amount of data to process and may be cumulative. Therefore, Processors are singleton. When multiple threads call into a Processor to do work, the Processor will block all incoming threads until the work is complete. Then, the Processor will release all of the threads that asked it to do work. This model allows all the analysis pipelines to share the same Processor without any duplication of work.

3.2.3 Analysis Pipelines

An Analysis Pipeline (also called Analysis Method) is the analytical center of the SPOONS framework. The pipeline aggregates multiple tasks that it needs to run on the data.

An Analysis Pipeline typically starts with running any number of processors on the data. Then, the pipeline invokes modelers on the data from the Processors. These modelers typically build models and predictive models on the data. Finally, the pipeline invokes tasks that asses the models produced in the previous step and decides whether or not there is an anomaly.

Every Analysis Pipeline gets its own thread, and there is no interdependence between the different pipelines. Currently, SPOONS usually runs more than 20 Analysis Pipelines at a time.

Tasks

Tasks are the core unit of computation in SPOONS. Almost everything that can be "run" is a child of the Task base class. Every Task gets its own thread, and callers into the Task may request that the task block the calling thread until the Task is complete.

Tasks are singleton with respects to the leaf child class. Therefore there are many tasks, but every task is unique. We do this by enforcing that the class name is unique upon construction. The uniqueness of tasks is very important to SPOONS distribution model that will be discussed in Section 3.2.5.

Modelers

Modelers are Tasks that are responsible for building a mathematical representation for the data.

Predictors Predictors build a predictive model of the data. For example, we have noticed that tweet volume tends to be periodic day-to-day and week-to-week. Therefore, a Predictor may just model that prediction by guessing that the volume in the future will be the same as it was the previous week or day.

Counters Counters attempt to build a model of data that was actually gathered by the system. Going with the previous example, the Counter for modeling tweet volume would simple count the number of tweets gathered for a period.

Monitors

Monitors take the models produced by the Predictors and Counters and compares them. The Monitors are responsible for making the final decision on about a period of time being anomalous.

Auto-Tuning

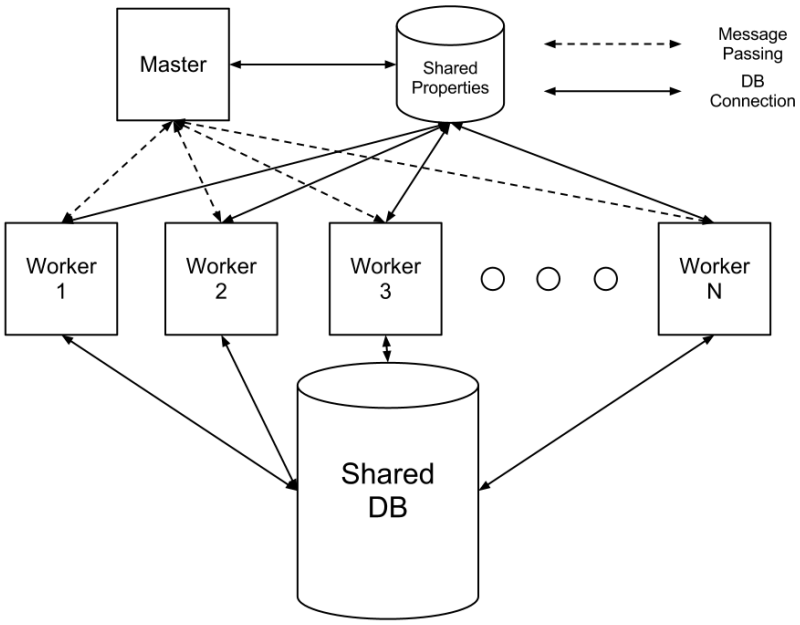

Figure 3.3: The server architecture of the SPOONS system.

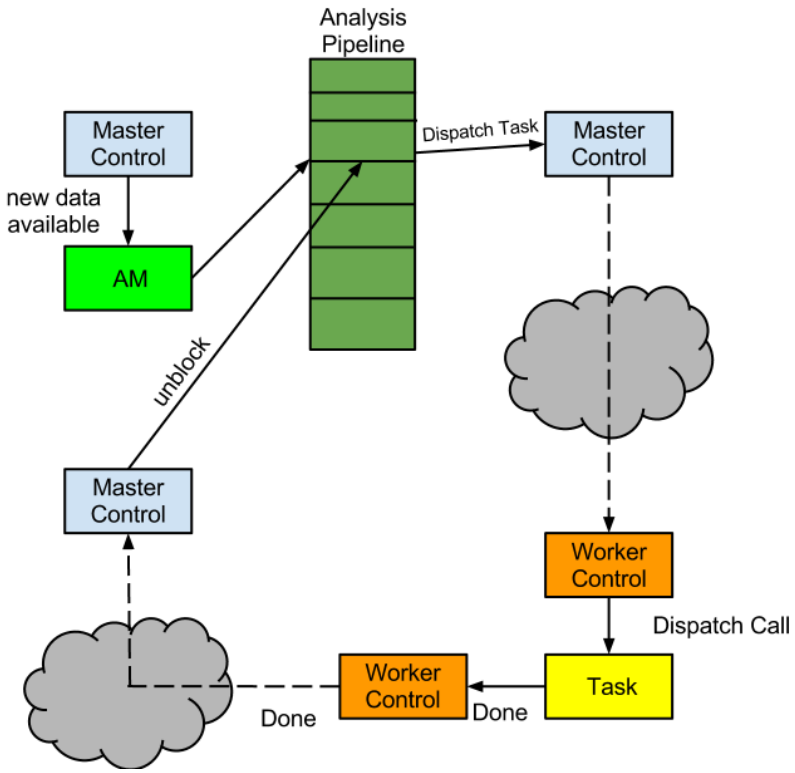

Figure 3.4: The control flow for distributable tasks.

3.2.4 Control

Master Control

Worker Control

Single Control

3.2.5 Distributed Model

Distributable Tasks

3.3 Database

3.3.1 Tables and Schemas

3.4 Analytical Framework

How do you plug into the framework. Instantiation based.

Chapter 4

Classifiers

4.1 Fitting Into The SPOONS Framework

4.2 WEKA Classifiers

4.3 Non-WEKA Classifiers

4.4 Feature Selection

4.4.1 Text Filtering

Twitter Specific Symbols

Emoticon Parsing

Title Replacement

Stemming

Perhaps not a ful subsubsection for this, just Porters

Stop Word Removal

4.5 Tweet Classes

4.5.1 Tweet Groups

4.6 Training Set

Chapter 5

Experimental Setup

5.1 Ground Truth

5.2 Success Metrics

5.3 Classifier Evaluation

5.4 Outage Detection

5.4.1 Monitor Parameters

Chapter 6

Results

6.1 Classifier Evaluation

6.2 Outage Detection

Chapter 7

Conclusions

7.1 Current Limitations of SPOONS

Severity Nature of Outage Malicious Tweet Attack Know What To Search For (dynamic search generation)

7.2 Current and Future Work

Kim - Advanced Sentiment Analysis Brett - Feasability of SPOONS as a comercial multi-target, source system. All sorts of stuff Open Source System Dynamic Training Set

Glossary

AWS Amazon Web Services. Cloud computing offerd by Amazon.

EC2 Elastic Compute Cloud. Instance based cloud computing machines offered through AWS.

Netflix Inc. [NASDAQ: NFLX] is the world's leading Internet subscription service for enjoying movies and TV series with more than 23 million streaming members in the United States, Canada, Latin America, the United Kingdom and Ireland[10].

Real Time Some of Netflix's services stream to customers in real time which means the users expect to get immediate responses from those services. So when they go down, the customers want the problem to be fixed immediately. These analysis methods need to have real time responses that are as close to immediate detection as possible. This means that the system needs to use whatever information it has available to it up to right before the outage to detect the event and alert Netflix engineers.

SPOONS Swift Perception Of Online Negative Situations. The name of the system presented in this paper.

Time Series Analysis The analysis of a series of data points over time. In this work those data points are the volume or estimated sentiment of a subset of the traffic about Netflix on Twitter during a time period.

Tweet A micro-post to a Twitter service. Tweets are limited to 140 characters.

Twitter Twitter is a social media service that allows users to post tweets (micro-posts) about any topic.

Bibliography

[1] N. Bansal and N. Koudas. Blogscope: a system for online analysis of high volume text streams. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 1410–1413. VLDB Endowment, 2007.

[2] A. Culotta. Detecting influenza outbreaks by analyzing twitter messages. In *KDD Workshop on Social Media Analytics*, 2010.

[3] M. Grinev, M. Grineva, A. Boldakov, L. Novak, A. Syssoev, and D. Lizorkin. Sifting micro-blogging stream for events of user interest. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 837–837, New York, NY, USA, 2009. ACM.

[4] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009.

[5] L. Hong and B. D. Davison. Empirical study of topic modeling in twitter. In *Proceedings of the First Workshop on Social Media Analytics*, SOMA '10, pages 80–88, New York, NY, USA, 2010. ACM.

[6] J. Huang, B. Zhou, Q. Wu, X. Wang, and Y. Jia. Contextual correlation based thread detection in short text message streams. *J. Intell. Inf. Syst.*, 38(2):449–464, Apr. 2012.

[7] F. Jabr. Using twitter to follow trends beats the stock market. *NewScientist*, (2829), Sept. 2011. http://www.newscientist.com/article/mg21128295.900-using-twitter-to-follow-trends-beats-the-stock-market.html.

[8] Y. Matsu, M. Okazak, and T. Sakak. Earthquake shakes twitter users: real-time event detection by social sensors. In *WWW 2010: Proceedings of the 19th World Wide Web Conference*, 2010. http://ymatsuo.com/papers/www2010.pdf.

[9] K. McEntee. personal communication, 2011.

[10] Netflix Inc. Netflix releases fourth-quarter 2011 financial results. *Netflix Press Release*, 2011. http://netflix.mediaroom.com/index.php?s=43&item=438.

[11] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, pages 79–86, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[12] Twitter. #numbers, Mar. 2011. http://blog.twitter.com/2011/03/numbers.html.

[13] Twitter. Terms of service, June 2011. https://twitter.com/tos.

[14] Twitter. Get search/tweets, Oct. 2012. https://dev.twitter.com/docs/api/1.1/get/search/tweets.