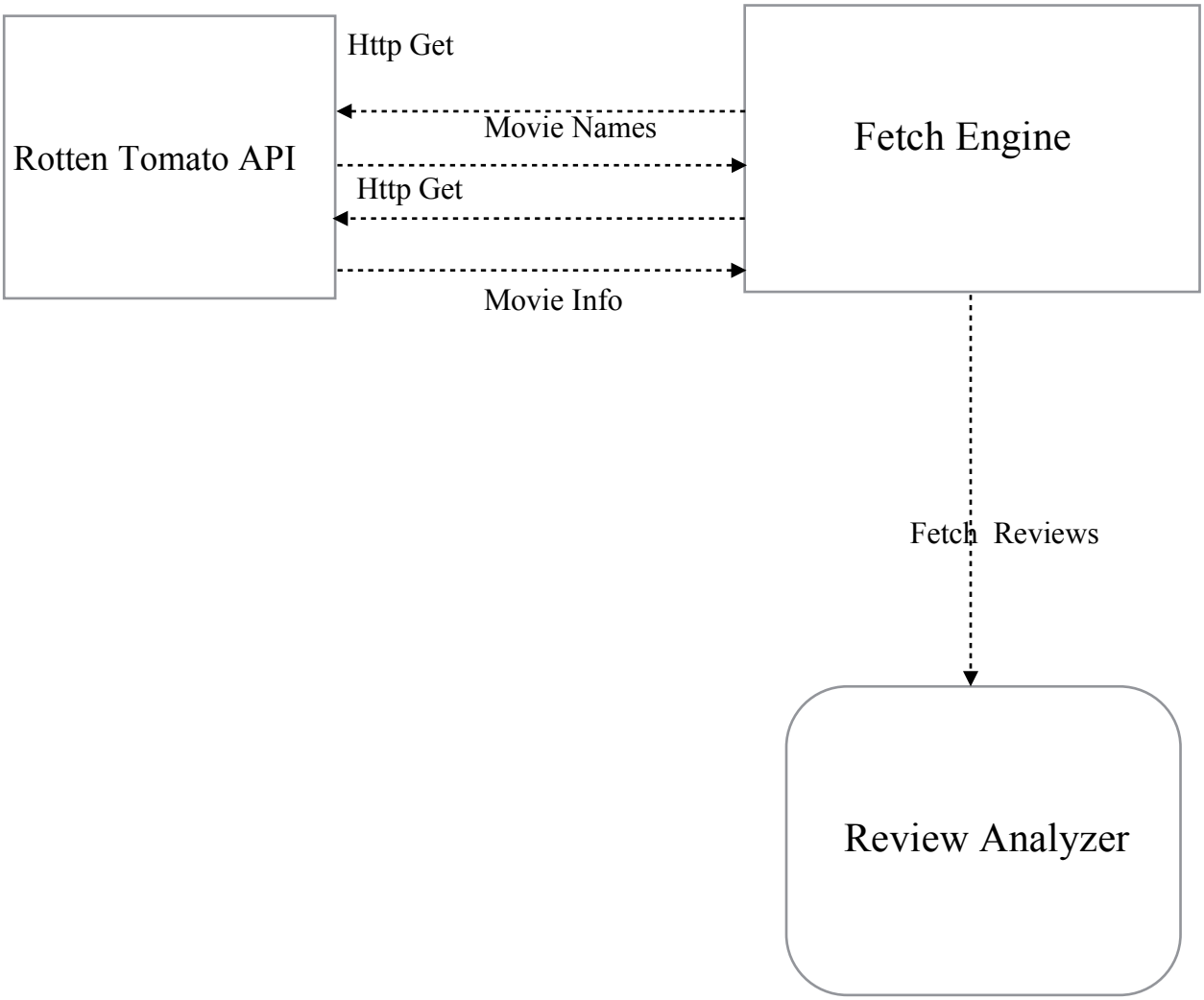
CS560 KDM Project Increment 2

By
Cuong Cu
Rishabh Bhojak
Vishnu Chelle
Ran Chen

- **System Architecture Diagram**



This is the general architecture of our application for Phase 2. The core elements are, Fetch Engine, Rotten Tomato & Analyzer.

1. Fetch Engine: Fetch engine is the program which makes the Http get request to get movie names from Rotten Tomato using their API, Once the fetch engine gets the movie names, using their movie ID it makes another Http request to get info of that movie. That's how we are getting our data.

2. Review Analyzer: This is our main Component, It fetches the data from fetch engine and process the data(movie reviews) to analyze it. It mainly consist of the following sub-components,

    1. Text Extractor: Once the review is fetched, text extractor runs on top of the review, to divide the sentences. Sentence with before any full stop(dot) is considered as a single sentence.

    2. NLP, Runner: The NLP, Runner looks for the key words after parses each sentence independently. We have been able to achieve the negation sentences and reference words.

    3. Coreference: We are using Stanford Coreference method for our application, which makes it smart to understand words like "it", "they", etc. This helps to understand the long reviews.

    4. Sentiment Analysis: Once the extractor extracts the review into sentences and NLP, Runner and Coreference works on these sentences the Sentiment Analysis do the classification, here we are

using 5 classes. (Very Negative, Negative, Neutral, Positive, Very Positive).

5. Scoring: Review Analysis does the Sentiment Analysis based on score, this scoring is done based on the key words for reviews and gives each sentence a score, from 0 to 4, where 0 is very negative and 4 is very positive.

- **Domain Model**

Data Sources: We are using Rotten Tomato, as our data source for phase 2, Using Rotten Tomato API we are fetching the reviews based on the name, the user selects and later, these reviews are analyzed. We are collecting maximum 50 reviews for each movie.

Here is the basic example to collect data using Rotten Tomato API:
Review Sample URL:

http://api.rottentomatoes.com/api/public/v1.0/movies/24220/reviews.json?review_type=top_critic&page_limit=50&page=1&country=us&apikey=zsnu54m8bqfzbct8ju4jz88g

Movie Search Sample URL:
http://api.rottentomatoes.com/api/public/v1.0/movies.json?q=king&page_limit=50&page=1&apikey=zsnu54m8bqfzbct8ju4jz88g

Algorithms used:

We have an algorithm similar to map reduce

Base forms of words

Sentence Extractor

Parts of Speech

Scoring

Noun for headlining every sentence

Sentiment Analysis

Coreference

Here is the pseudocode which does the mapper & reducer job in our application

```
mapping (Coref coref):
   topic := getTopic (coref)

   for sentence in getSentences (coref):
      emit (topic, sentence)

reducing (String topic, Iterator sentences):
   S := []

   for sentence in sentences:
      append (S, sentence)

   emit (topic, S)

reviewAnalyze (String review):
   C := StanfordNLP.getCorefs (review)

   M := {}
   for coref in C:
      put (M, mapping (coref))

   R := reducing (M)

   for topic in R:
      emit (topic, StanfordNLP.sentimentAnalyze (R[topic]))
```

Glue Logic(Mashup)

Using Rotten Tomato to get movie reviews, using CoreNLP extractor from Stanford to use the algorithms and using coreference with Senti-

ment Analysis to understand the reviews with words like, “it”, “this”, etc for actor, movie names, etc.

Features:

1. Understanding Context/Topic of every sentence

2. Calculating score for every sentence based on words & meaning

3. Coreference

4. Context based Sentiment/Opinion analysis for every sentence.

Expected Inputs/Outputs

Input: Movie Reviews (Upto 50)

Output: Context Based Sentiment/Opinion Analysis, for every sentence.

Scoring (0 - 4)

0: Very Negative

1: Negative

2: Neutral

3: Positive

4: Very Positive

Example:

Input:
Tim Burton's Batman wasn't a bad movie. It was entertaining and had a nice comic-book feel to it, but it struck me as being quite formulaic. Some scenes worked for me, but others felt forced and oddly out of place. I imagine Tim Burton had some of the acto

Output:


```
"{
  "Tim Burton": [
    {
      "score": 1,
      "opinion": "Tim Burton's Batman wasn't a bad movie."
    },
    {
      "score": 0,
      "opinion": "I imagine Tim Burton had some of the actors, like Jack Palance, overact on purpose, to give the movie a similar feel to the Batman TV show from the 1960s; however, the result felt more like a high-school play than a movie."
    }
  ],
  "Jack Palance": [
    {
      "score": 0,
      "opinion": "I imagine Tim Burton had some of the actors, like Jack Palance, overact on purpose, to give the movie a similar feel to the Batman TV show from the 1960s; however, the result felt more like a high-school play than a movie."
    }
  ],
  "the movie": [
    {
      "score": 2,
      "opinion": "It was entertaining and had a nice comic-book feel to it, but it struck me as being quite formulaic."
    },
    {
      "score": 1,
      "opinion": "Tim Burton's Batman wasn't a bad movie."
    },
    {
      "score": 0,
```

"opinion": "I imagine Tim Burton had some of the actors, like Jack Palance, overact on purpose, to give the movie a similar feel to the Batman TV show from the 1960s; however, the result felt more like a high-school play than a movie."
}
]
}"

- **Application Specification**

System Specification:

IntelliJ IDEA IDE

Java 1.6 or higher

CoreNLP libraries

Play framework

Languages used:

Scala

Java

General flow of our application

User provide a movie name -> list all possible movies on web page -> user chooses a movie from suggested list of movie -> display 50 first reviews of the selected movie.

Implementation:

Algorithms used,

We have an algorithm similar to map reduce

Base forms of words

Sentence Extractor

Parts of Speech

Scoring

Noun for headlining every sentence

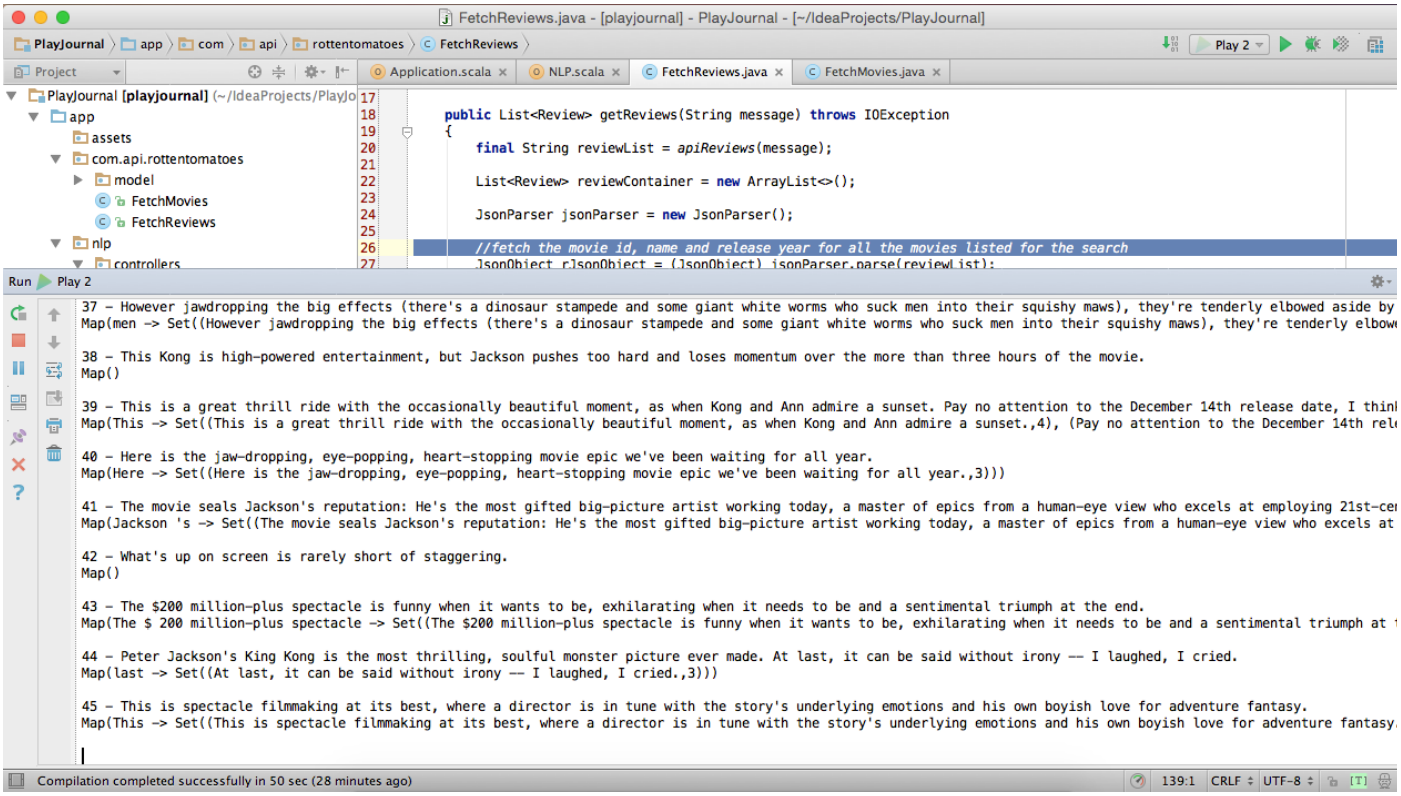
Sentiment Analysis

Coreference

We have implemented all of them. We are looking to add more features in the future , for instance, recommendation system, etc.

Currently we do not have any visualization or web application of our application, we are currently working on it.

- **Documentation**



As seen in the screenshot, it shows for 45 reviews collected from Rotten Tomato for the movie King Kong.
Here you can see, all the reviews are extracted and sentence by sentence parsing is done. Lets take one example here, 45th review. This is refered to the set. And the review about this reference is very positive. (4 score)

We follow Batch-Sequential architecture:
Cause of its nature: simplicity
Design Flow
Review From Rotten Tomato using their API ->
Mapper -> Reducer ->
Analyzer -> Results

Project Management:

Implementation status report

Work: Rotten Tomato API

To get movie names, movie info and movie reviews based on the names selected by user. We used java code to implement the same.

Time Taken: 4 Hours

Contributions: Vishnu Chelle(50), Ran Chen(50)

Work: Coreference

To use Standard Coreference in our application to understand words like “this”, “that”, etc. We used Scala to implement the same.

Time Taken: 10 Hours

Contributions: Cuong Cu(60), Rishabh Bhojak(40)

Work: CoreNLP

To use CoreNLP in our application using Stanford libraries, which laid the core work of the application. We used scala to implement the same.

Time Taken: 8 Hours

Contributions: Cuong Cu(30), Ran Chen(20), Vishnu Chelle(25), Rishabh Bhojak(25)

Work: Integration of different algorithms used & General Testing

Integrating different algorithms and data sources combined to make the application as a whole. Also, we did a general testing of our application.

Contributions: Ran Chen(30), Vishnu Chelle(40), Rishabh Bhojak(15), Cuong Cu(15)

Work: Report Writing, miscellaneous work & issues fix

Writing the Report & miscellaneous work like GitHub, research, ideas, etc.

Rishabh Bhojak (60)

Vishnu Chelle (30)

Ran Chen (10)

Work: Issues/Bugs fixes

General issues faced by different members and fixed

Cuong Cu (60)

Vishnu Chelle(15)

Rishabh Bhojak (15)

Ran Chen (10)

Deployment:

Code:
https://github.com/cnsgcu/Stage02

Documentation:

https://github.com/rbx44/CS560

Bluemix:

We were not able to deploy our application on Bluemix as free users. But we can show a live demo of our application in person.