Opinion monitoring and analysis in social networks

Maria Khrustaleva
Peter the Great St. Petersburg Polytechnic University
maria-crystal78@yandex.ru

ABSTRACT

The Web application for analyzing users opinions in social networks is developed in the thesis. The Spring framework was used as a structure of the program system. The Twitter microblog was chosen as a social network. The purposes of the carried out work are following:

- information research from open sources;
- creation of the statistical tool used in the field of a business analytics for understanding of the current situation;

Testing and approbation of the developed solution was made on the real data.

Keywords

Web-application, Twitter, Spring MVC, scalable architecture, programming

1. INTRODUCTION

Social networks and blogs became very popular means of communication nowadays. Millions of users write their opinion about various aspects of everyday life. For this reason such services are very important data sources for the extraction and analysis of opinions. Having appeared and rapidly developed rather recently (first of all it's Twitter microblogs), these systems helped to discover more and more tools for the analysis of their content.

Text information can be divided into two main categories: facts and opinions. The facts are objective statements about some entities or events. Opinions are the subjective statements reflecting the person's relation or perception of some event. The bulk of existing researches about natural language processing are concentrated on factual information collection and extraction. It's traditional information search, web search - in particular. But there haven't been enough researches in case of opinions handling. Though the accounting of people's opinions (subjective text information) is very useful not only for individuals, but also for the organizations.

With the emergence and development of the Internet, and, as a result, with explosive growth of content created by its users new opportunities of information distribution and consumption were opened. There was an opportunity to publish feedbacks about products in online stores and to express the point of view concerning almost any things in various Internet-forums, blogs and social networks. As a result of all above-mentioned types of communication, the content volume created by users was increased. Now the consumers intending to make a purchase can find a huge number of the reviews and feedbacks estimating any goods in the network. When the organization wants to understand the relation of consumers concerning its products or services, traditionally, consultants are hired, surveys are conducted or focus groups are created by this organization. It could be avoided with tools for analysis of opinions in the network community. However the monitoring of sources with opinions is still a difficult task. There are a lot of various forums, blogs, etc. in the network, which contains a huge amount of information while the number of opinions in them on some specific question can be rather small. It's almost impossible to do some operations manually: to process such data arrays, to find and take necessary opinions from there, to distinguish their tonality, to generalize and lead result to a convenient form. Thus, the system of automatic collection and analysis of opinions is required.

2. METHODOLOGY

The application architecture was designed using the Model-view-controller approach (MVC, "model-view-controller"). It is a software design pattern for implementing user interfaces on computers. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user. This design pattern is often used for creation of an architectural frame when you pass from the theory to implementation in the specific data domain.

The main application objective of this concept is the separation of business logic (model) from its visualization (view). Due to such division the possibility of reuse raises. As with other software architectures, MVC expresses the "core of the solution" to a problem while allowing it to be adapted for each system.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

The central component of MVC, the model, captures the behavior of the application in terms of its problem domain, independent of the user interface.

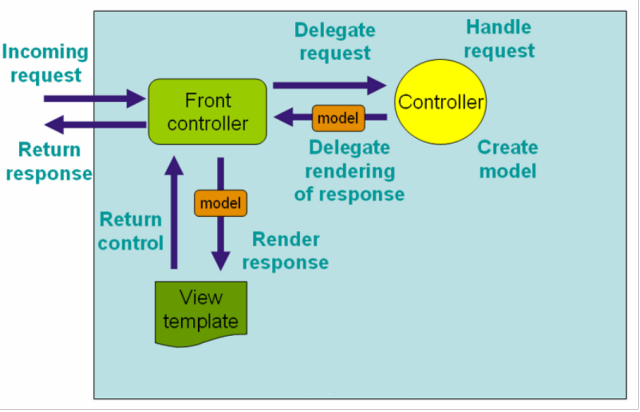
- The model directly manages the data, logic, and rules of the application.
- A view can be any output representation of information, such as a chart or a diagram. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.
- The third part, the controller, accepts input and converts it to commands for the model or view.

In addition to dividing the application into three kinds of components, the model-view-controller design defines the interactions between them.

- A model stores data that is retrieved according to commands from the controller and displayed in the view.
- A view generates new output to the user based on changes in the model.
- A controller can send commands to the model to update the model's state (e.g., editing a document). It can also send commands to its associated view to change the view's presentation of the model (e.g., by scrolling through a document).

In the implementation of the required web service each request is intercepted by the global Front-controller which determines by specific parameters (URI) to what controller to transfer the received request. The controller processes the request and creates model. The Front-controller fills the view with the model data and returns the received result to the browser. The diagram of the request processing in Spring on the picture 1 is given below.

Figure 1: The diagram of the request processing in Spring



The web service was implemented using the classical algorithm called Word Count. The task is formulated as follows: there is a large number of tweets. We check each tweet for the existence of the specific keyword: we look for an index i of the first entrance of substring in a tweet line - if an index non-negative, then we increment the counter. Further we look for an index of the following keyword appearance since the i+1 item, incrementing the counter in case of success. After analyzing all the tweet, the number of keyword detection in it is added with its quantity in the previous already viewed tweets. Twitter has a restriction of requests amount because of probability of the excessive load on their servers. Therefore the solution with data cached in the memory was developed.

3. EXPERIMENT

The developed application obtains input data: the number of accounts for the analysis and the list of keywords. The Twitter programming interface is called to select data. So the request redirects to the Twitter API. Further the subscribers of the fixed user are boot. The user of Barack Obama was used because of the big amount of his subscribers.

Accidental subscribers are selected, all their tweets are booted by 200 pieces per call since the Twitter API has a limit on a single request of the messages boot. Then all the tweets are being analysed on existence of the entered keywords with the Word Count algorithm. As a result the pie chart with analysis results is displayed.

For creation of a web service the following units were designed and implemented:

1. **The REST-client for connection to a backend**

   The REST client is filled automatically with Spring framework, using settings for the appendix where the backend address is specified. The client sends the HTTP GET request on the backend service URI with the parameters entered by the user - number of accounts for the analysis and keywords. Also the client receives the response in the JSON format, transferring it to a collection with words and their quantity among the analyzed accounts.

2. **Web-service controller (Front-controller)**

   This is the main controller of a Spring MVC framework. It is loaded with the start of the application and automatically sets the URI whereby the REST client will be connected for communication with the backend. When the web-service is started the client locates at the address "/" - in this case Front-controller won't receive data from a backend therefore the empty request is used. After entering of input parameters by the user and sending the form with these data the client is redirected to the address "/load-request". Then the controller receives a request for the analysis, transfers it to the backend, takes the received model and fills the view with data received, builds the chart by results.

3. **Working service controller (WorkerController)**

   This controller receives the number of accounts which are had to be analyzed, and the keywords coming from the Front-controller after the REST-client query with parameters. The controller loads tweets by 200 pieces per call, reads words and gives result to the web, responds in the json format.

4. RESULTS AND DISCUSSION

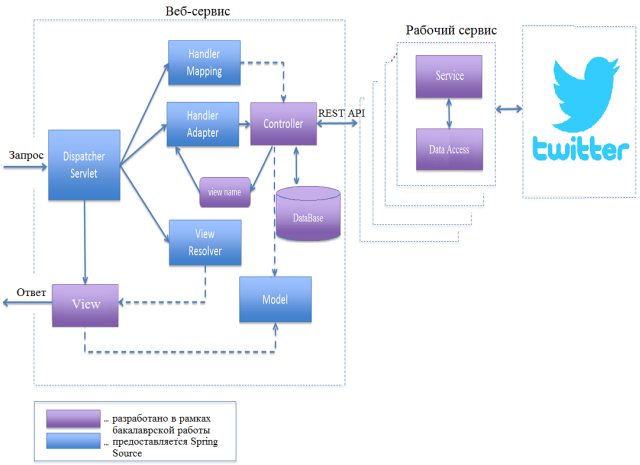
Approbation of the developed web-application on real data was carried out. Several brands from one category were

chosen for their handling by the service. The number of references of each brand in "interviewed" users tweets was returned at the exit. Thus, this service makes the comparative analysis of brands popularity in the microblog that corresponds to a state of affairs in a real life.

Also operation performance of application was checked. The different number of the analyzed accounts was entered. The maximum number of users, whose tweets were used, was 500 until the program fell. It's not enough for such a service. But the number of successfully processed information sharply increases after deployment the service in the cloud infrastructures.

The request lifecycle of the developed application is presented on the picture 2.

Figure 2: The request lifecycle of the developed application



5. CONCLUSION

Within this thesis the following results were received:

- The program system performing the analysis of messages in the microblog and issuing the report in graphical representation is created.
- The scaleable architecture for deployment of service in cloud infrastructures is constructed.
- Testing and approbation of web-service on real data is held.

Efficiency of using a design pattern in case of which the application was divided into three components was shown. It allows to create scaleable architecture for handling large volumes of data.

Further it is planned to realize an algorithm of thenbrands assessment depending on emotional coloring of users messages.