International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 14
October 2016

On Summarization and Timeline Generation for Evolutionary Tweet Streams

U. Ajay
DEPARTMENT OF CSE
MADHIRA INSTITUTE OF TECHNOLOGY AND SCIENCES

G L Chandrasekhar,Assistant Professor
DEPARTMENT OF CSE
MADHIRA INSTITUTE OF TECHNOLOGYAND SCIENCES

ABSTRACT:

Short-text messages such as tweets are being created and shared at an unprecedented rate. Tweets, in their raw form, while being informative, can also be overwhelming. For both end-users and data analysts, it is a nightmare to plow through millions of tweets which contain enormous amount of noise and redundancy. In this paper, we propose a novel continuous summarization framework called Sumblr to alleviate the problem. In contrast to the traditional document summarization methods which focus on static and small-scale data set, Sumblr is designed to deal with dynamic, fast arriving, and large-scale tweet streams. Our proposed framework consists of three major components. First, we propose an online tweet stream clustering algorithm to cluster tweets and maintain distilled statistics in a data structure called tweet cluster vector (TCV). Second, we develop a TCV-Rank summarization technique for generating online summaries and historical summaries of arbitrary time durations. Third, we design an effective topic evolution detection method, which monitors summary-based/volume-based variations to produce timelines automatically from tweet streams. Our experiments on large-scale real tweets demonstrate the efficiency and effectiveness of our framework.

INTRODUCTION

INCREASING popularity of microblogging services such as Twitter, Weibo, and Tumblr has resulted in the explosion of the amount of short-text messages. Twitter, for instance, which receives over 400 million tweets per day1 has emerged as an invaluable source of news, blogs, opinions, and more. Tweets, in their raw form, while being informative, can also be overwhelming. For instance, search for a hot topic in Twitter may yield millions of tweets, spanning weeks. Even if filtering is allowed, plowing through so many tweets for important contents would be a nightmare, not to mention the enormous amount of noise and redundancy that one might encounter. To make things worse, new tweets satisfying the filtering criteria may arrive continuously, at an unpredictable rate.

One possible solution to information overload problem is summarization. Summarization represents a set of documents by a summary consisting of several sentences. Intuitively, a good summary should cover the main topics (or subtopics) and have diversity among the sentences to reduce redundancy. Summarization is extensively used in content presentation, specially when users surf the internet with their mobile devices which have much smaller screens than PCs. Traditional document summarization approaches, however, are not as effective in the context of tweets given both the large volume of tweets as well as the fast and continuous nature of their arrival. Tweet summarization, therefore, requires functionalities which significantly differ from traditional summarization.

In general, tweet summarization has to take into consideration the temporal feature of the arriving tweets. Let us illustrate the desired properties of a

tweet summarization system using an illustrative example of a usage of such a system. Consider a user interested in a topic-related tweet stream, for example, tweets about "Apple". A tweet summarization system will continuously monitor "Apple" related tweets producing a real-time timeline of the tweet stream. As illustrated in in this system, a user may explore tweets based on a timeline (e.g., "Apple" tweets posted between October 22nd, 2012 to November 11th, 2012). Given a timeline range, the summarization system may produce a sequence of times tamped summaries to highlight points where the topic/subtopics evolved in the stream. Such a system will effectively enable the user to learn major news/ discussion related to "Apple" without having to read through the entire tweet stream.

IMPLEMENTATION

Admin

In this module, the Admin has to login by using valid user name and password. After login successful he can do some operations such as search history, view users, request & response, all topic messages and topics.

Search History

This is controlled by admin; the admin can view the search history details. If he clicks on search history button, it will show the list of searched user details with their tags such as user name, searched user, time and date.

Users

In user's module, the admin can view the list of users and list of mobile users. Mobile user means android application users.

Request & Response

In this module, the admin can view the all the friend request and response. Here all the request and response will be stored with their tags such as Id, requested user photo, requested user name, user name request to, status and time & date. If the user accepts the request then status is accepted or else the status is waiting.

SYSTEM STUDY

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement,

as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

SOFTWARE ENVIRONMENT

Java Technology

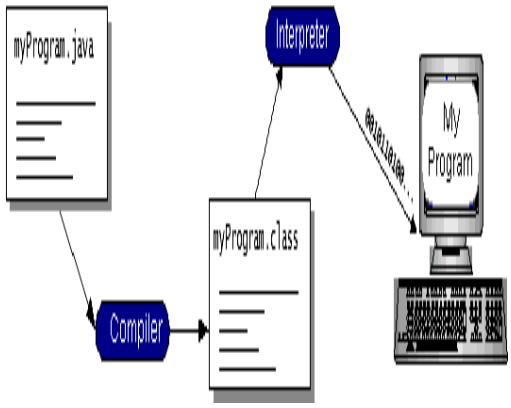
Java technology is both a programming language and a platform.

The Java Programming Language

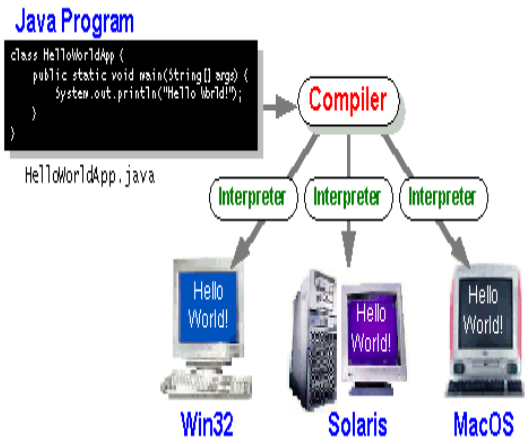
The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded
- Robust
- Dynamic
- Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes* —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



You can think of Java byte codes as the machine code instructions for the *Java Virtual Machine* (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



What Can Java Technology Do?

The most common types of programs written in the Java programming language are *applets* and *applications*. If you've surfed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

SYSTEM SPECIFICATION

Hardware Requirements:

- System : Pentium IV 3.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 1.44 Mb.
- Monitor : 14' Colour Monitor.
- Mouse : Optical Mouse.
- Ram : 1 GB.

Software Requirements:

Operating system : Windows Family.

- Coding Language : J2EE (JSP,Servlet,Java Bean),Android 4.4
- Data Base : MY Sql Server.
- IDE : Eclipse Juno
- Web Server : Tomcat 6.0

SYSTEM TESTING

TYPES OF TESTS

Integration testing
Functional test
White Box Testing
Black Box Testing

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications

CONCLUSIONS

We proposed a prototype called Sumblr which supported continuous tweet stream summarization. Sumblr employs a tweet stream clustering algorithm to compress tweets into TCVs and maintains them in an online fashion. Then, it uses a TCV-Rank summarization algorithm for generating online summaries and historical summaries with arbitrary time durations. The topic evolution can be detected automatically, allowing Sumblr to produce dynamic timelines for tweet streams. The experimental results demonstrate the efficiency and effectiveness of our method. For future work, we aim to develop a multi-topic version of Sumblr in a distributed system, and evaluate it on more complete and large-scale data sets.

REFERENCES

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for\ clustering evolving data streams," in Proc. 29th Int. Conf. Very Large Data Bases, 2003, pp. 81–92.

[2] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 1996, pp. 103–114.

[3] P. S. Bradley, U. M. Fayyad, and C. Reina, "Scaling clustering algorithms to large databases," in Proc. Knowl. Discovery Data Mining, 1998, pp. 9–15.

[4] L. Gong, J. Zeng, and S. Zhang, "Text stream clustering algorithm based on adaptive feature selection," Expert Syst. Appl., vol. 38, no. 3, pp. 1393–1399, 2011.

[5] Q. He, K. Chang, E.-P. Lim, and J. Zhang, "Bursty feature representation for clustering text streams," in Proc. SIAM Int. Conf. Data Mining, 2007, pp. 491–496.

[6] J. Zhang, Z. Ghahramani, and Y. Yang, "A probabilistic model for online document clustering with application to novelty detection," in Proc. Adv. Neural Inf. Process. Syst., 2004, pp. 1617–1624.

[7] S. Zhong, "Efficient streaming text clustering," Neural Netw., vol. 18, nos. 5/6, pp. 790–798, 2005.

[8] C. C. Aggarwal and P. S. Yu, "On clustering massive text and categorical data streams," Knowl. Inf. Syst., vol. 24, no. 2, pp. 171–196, 2010.

[9] R. Barzilay and M. Elhadad, "Using lexical chains for text summarization,"
in Proc. ACL Workshop Intell. Scalable Text Summarization, 1997, pp. 10–17.

[10] W.-T. Yih, J. Goodman, L. Vanderwende, and H. Suzuki, "Multidocument
summarization by maximizing informative contentwords," in Proc. 20th Int. Joint Conf. Artif. Intell., 2007, pp. 1776–1782.

[11] G. Erkan and D. R. Radev, "LexRank: Graph-based lexical centrality as salience in text summarization," J. Artif. Int. Res., vol. 22, no. 1, pp. 457–479, 2004.

[12] D. Wang, T. Li, S. Zhu, and C. Ding, "Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization," in Proc. 31st Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2008, pp. 307–314.