Chat Message Analysis

Theekshana M.A.H

(IT 14 0299 50)

Degree of Bachelor of Science

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

October 2017

Chat Message Analysis

Project Title: Chat Review
Project ID:16-066

Supervisor: Dr. Darshana Kasthurirathne

Dissertation submitted in partial fulfillment of the requirements for the degree of Science

Department of Information Technology

Sri Lanka Institute of Information Technology

October 2017

DECLARATION

I declare that this is my own work and this dissertation1 does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to Sri Lanka Institute of Information Technology the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

………………………………… …../…../……….

Theekshana M.A.H

IT 14 0299 50

The above candidate has carried out research for the B.Sc. Special (Hons) degree in IT Dissertation under my supervision.

………………………………… …../…../……….

Dr. Darshana Kasthurirathne

(Signature of the Supervisor)

ABSTRACT

The use of Internet chat applications has benefited many different segments of society. It also creates opportunities for criminal enterprise, terrorism, and espionage. We present a study of a real-world application of chat analysis which will analyze chat messages in four ways such as topic detection, Emotion Extraction, evaluate healthy and Personal information sharing analysis. Also, analyzing chat traffic has important applications for both the military and the civilian world. Here on this document, it compares the results of an unsupervised learning approach with those of a supervised classification approach with regards to chat review application. The paper also discusses some of the specific challenges presented by this chat review application.

Unsupervised learning techniques such as clustering are very popular for analyzing text for topic identification as well as emotion extraction. These techniques have several attractive features, the most significant being that they do not require labeled training examples. This however is also a disadvantage under some circumstances. Therefor meantime we do this research we will discover more and more technologies required for analyzing chat massages based on four different categories such as topic detection, Emotion Extraction, evaluate healthy and Personal information sharing analysis.

With use of this chat analysis application user will be able identify the chatting partner in analytical way. And system will keep an analytical review for each chat session user interacted. Also, system will be capable of showing its analytical data in a user friendly manner (in a graphical way).

ACKNOWLEDGEMENT

The work described in this research paper was carried out as our 4th year research project for the subject Comprehensive Design Analysis Project. The completed final project is the result of combining all the hard work of the group members and the encouragement, support and guidance given by many others. Therefore, it is researchers' duty to express their gratitude to all who gave them the support to complete this major task.
The researchers are deeply indebted to supervisor Dr. Darshana Kasthurirathne and Lecturers of Sri Lanka Institute of Information Technology whose suggestions, constant encouragement and support in the development of this research, particularly for the many stimulating and instructive discussions. The researchers also wish to thank all colleagues and friends for all their help, support, interest and valuable advices. Finally, the researchers would like to thank all others whose names are not listed particularly but have given their support in many ways and encouraged researchers to make this a success.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

1 INTRODUCTION

With the ever-increasing use of the Internet, computer-mediated communication via textual messaging has become popular. This type of electronic discourse is observed in point-to-point or multicast, text-based online messaging services such as chat servers, discussion forums, emails and messaging services, newsgroups, and IRCs (Internet relay chat). These services constantly generate large amounts of textual data, providing interesting research opportunities for mining such data. We believe that extracting useful information from this kind of messages/conversations can be an important step towards improving the human–computer interaction

Specifically, machine learning can be a powerful tool for analyzing electronic discourse data. This work particularly concentrates on the data obtained from chat servers, which provide a point to-point online instant messaging facility over the Internet. We investigate the rate of success in the problem of predicting various author- and message-specific attributes in chat environments using weka library's. For this purpose, we first employ a term-based approach and formulate the chat mining problem as an automated text classification problem, in which the words occurring in chat messages are used to predict the attributes of the authors (e.g., age, gender) or the messages (e.g., the time of a message). Second, we employ a style-based approach and investigate the effect of stylistic features (e.g., word lengths, use of punctuation marks) on prediction accuracies, again for both author and message attributes. Finally, we briefly discuss the effect of the author and message attributes on the writing style

The main contributions of this study are four-fold. First, the chat dataset used in this work has unique properties: the messages are communicated between two users; they are unedited; and they are written spontaneously. We believe that extracting information from real-time, peer-toper, computerized messages may have a crucial impact on the areas such as financial forensics, threat analysis, and detection of terrorist activities in the near future. Our work presents a new effort in that direction, aiming to retrieve previously unexplored information from computerized communications. Second, for the first time in the literature, several interesting attributes of text and its authors are examined. Examples of these attributes are educational affiliations and connectivity domains of the authors and the receivers of the messages. Third, the performance of term- and style-based feature sets in predicting the author and message attributes are compared via extensive experimentation. Fourth, to the best of our knowledge, our work is the first one that investigates real-time, peer-to-peer, computerized communications in the context of authorship studies. Our findings are good pointers for researchers in this new application area, namely Evaluate healthy of particular chat session.

1.1 Background Context

The goal of executing this research project is to build up a chat messages analysis web application, identify characteristics of the chatting partner by analyzing chat messages.
Identify characteristics of the chatting partner by analyzing chat messages. Nowadays we meet strangers all the time in our day to day life. So, we attempt to have partnership with them without any fear at all. Also, some time they are more and more smart than we think, then it's very difficult to identify the characteristics by only looking at their messages, because any one can send anything. Nowadays it is highly required to have intelligent way to detect those frauds (may be what that message really mean). Solution is to use analytical application for online chatting. Then we can review our messages in analytical way or it will lead us to think about our chatting partner in analytical way. main objective of the application is simply gives us clear idea about the message by analyzing it in various ways. In that case, we can be very much aware about our chatting partner. So, this application let you deal with that strange people in understandable way. It shows you auto generated graphs based on emotion, personal information sharing and topics discussed etc. And also, we can confirm the outcomes of this system by analyzing public profile, and other related social profiles of particular user or other data sources if possible

Chat is an increasingly important form of CMC (Computer-mediated communication). It is employed by many sectors of society to improve communication, create value, and commit crimes. In this chapter, we explore chat first as it relates to other human language modalities. Then, we explore Natural Language Processing (NLP) and its goals, followed by its applicability to chat. Finally, we discuss the idea of topicality and previous Machine Learning (ML) techniques used to detect topics in chat. The objective of automatically revealing the topic of any form of communication is twofold. The first motive is to increase the knowledge of how humans communicate, to unravel the mystery of information conveyance. The second is to build useful systems. Topic detection and emotion extraction in this context are steps toward automating tasks that would otherwise be untenable, because the sheer volume of data makes it impractical. Section A provides working definitions of CMC, chat, and natural human languages

1.2 Research Gap

The system which is named as “chat review for Evaluate healthily of particular chat sessions” is proposed to develop as a solution for a potential way of integrating an intelligent SCRM with Social media chat messages like Facebook , what’s app. Business today is increasingly focusing on services through social media and it has been getting more attention, in line with this trend, designing CRM effectively and efficiently shed light on the connection between CRM and the social media, since they want to improve their performance efficiently.

Research will investigate the problem to find what extent CRM can be inferred from the chat analysis and to illustrate the problem, proposed system will analysis customer messages through provide services for their requests at real time by using the Natural Language Processing technology (NLP).

Other than that, the system extracted data from the social media will be categorized automatically. The information can be general knowledge based information regarding the customer messages which is provided from above mentioned social media channels. In information retrieval, Natural language processing will come to play a handy role to understand the user queries which will be in Natural Language (English). So, when the queries submitted by the user it will pass to the core system and get categorized to the appropriate s/w or h/w category. Agents will get the notification mentioning that they got a question for their category. Agents can answer the questions using text chat or any other way user preferred. Natural Language Processing ability as well as semantic information storage capabilities make the proposed system unique as well as efficient. These capabilities will allow agent to represent and understand the user provided information meaningfully like the way humans does

The “Chat review” is a web based application which helps peoples or companies to analysis their chatting information’s. the application should be free to download from either pc application store or similar services.

In my research area “evaluate healthy of particular chat” particular chat message information will category’s three ways.

- ✓ Extract fewer first – person pronouns.
- ✓ Fewer Exclusionary words.

- ✓ Unusual details from the collected messages.

The Objective is to design a flexible intelligent, efficient and real time SCRM system and it's a kind of a cost effective, time saving way to a profitable business with new technologies.

1.3 Research problem

The chat evaluate healthy messages problem can be considered as a single-label classification problem. If the attribute to be predicted is user-specific, a supervised learning solution to this problem is to generate a prediction function, which will map each user instance onto one of the attribute classes. The prediction function can be learned by training supervised classification algorithms over a riper sensitive set of user instances whose attributes are known. In case of message-specific attributes, the process is similar. However, this time, the individual chat messages are the instances whose attributes are to be predicted, and the training is performed over a set of chat messages whose attributes are known.

In predicting the user-specific attributes, each user instance is represented by a set of features extracted from the messages that are generated by that particular user. Similarly, in predicting message-specific attributes, each message instance is represented by a set of features extracted from the message itself. In this work, for predicting both types of attributes, we evaluate two competing types of feature sets: term-based features versus style-based features. When term-based features are used, the vocabulary of the message collection forms the feature set, i.e., each term corresponds to a feature. In predicting user-specific attributes, the set of terms typed by a user represents a user instance to be classified. In predicting message-specific attributes, the terms in a message represent a message instance. This type of a formulation reduces the chat mining problem to a standard text classification problem

In literature, term-based feature sets are widely used (Lam, Ruiz, & Srinivasan, 1999). Unfortunately, term-based features may not always reflect the characteristics of an author since the terms in a document are heavily dependent on the topic of the document. In chat mining, a feature set that is independent of the message topic may lead to better results in predicting the userand message-specific attributes. Hence, using the stylistic preferences instead of the vocabulary emerges as a viable alternative.

Rudman (1998) states that there are more than 1000 different stylistic features that can be used to define the literary style of an author. The most commonly used stylistic features are word frequencies; sentence and word lengths; and the use of syllables, punctuation marks, and function words (Holmes, 1985). So far, there is no consensus on the set of the most representative features.

This study, in addition to the traditional stylistic features, considers several new and problem specific stylistic features (e.g., smileys and emoticons) used in order to find better representations for user or message instances. The smileys and emoticons are two important features that are

1.4 Research Questions

Main function of my research part is evaluate healthy messages of total chat session and show graphical format.

Main questions faced are,

- How to evaluate the users chat session and show the unused full messages, total messages.
- Fewer Exclusionary words.
- Extract fewer first – person pronouns

And I had a huge difficulty to connect the data set to PHP because each and every library's and algorithm developed by java and after we convert into .api file and its connect into PHP.

1.5 Research objectives

At the heart of any business are customers and the proposed system with social networking represents an opportunity to build even more mutually rewarding and candid relationships with those customers and create a bond.

The general objectives are

- ➢ The main objective is to develop a web application integrated with social media which will enable customers to have services efficiently at real-time by being in a social chat analysis.
- ➢ The secondary objective is to design and build the project to achieve non-functional requirements such as response time, reliability, usability, supportability etc.

Specific objectives are

- Work better, together - **tracking and filtering customer messages**.
- Services Sell in the new social era- **tracking and filtering customer messages, storing and retrieving knowledge**
- Publish fresh content in graphical format-**Handling user chat profiles**
- Engage everyone with social experiences-**View Dashboards**

Mainly this system is focus on two different users who will get benefits from the chat analysis application will be:

1. social media users who can identify to the problems of extract topic including each chat

2. Business people who expect indirect flow of money for their business

2 METHODOLOGY

2.1 Methodology

Several methodologies were used in the developing process by the development team. In my research part, Weka model have been used. At the beginning of the development of the system, a deep study about socializing concept was carried out and different articles were followed based on those. Weaknesses in the existing systems and new requirements were identified after communicating with different people with different age levels. Nowadays at least one social media application is being used by people belong to each and every age group. So the best way to do a study was to get some ideas and feedbacks from those users. At present social media applications are being used for different purposes. Therefor different users are having different perspectives. After the study, consideration of physical proximity was identified as the most important part missing in the existing applications.

After all the studies that had been carried out, I came up with a new area for socializing these applications. Displaying online friends of a particular user inside a preferred radius was considered as one major segment. Radius can be selected by the user. The reason why this is important is it's more practical and it will be more useful. As I mentioned above, if a friend of a user who's not in a close proximity wanting a ride from the user would be pointless. Friends who are not even in the country is being displayed in the map by existing applications which is not a practical scenario and identified as a major drawback of those applications.

Technologies and application platforms that we were going to use were decided at the important initial stages. Technical difficulties of the development methodologies that we were going to use and the efficiency of the algorithms we were going to implement were examined after doing a deep literary survey and surfing through internet. In the circumstances like ours, doing discussions and surveys is considered as the best or the only way to solve such technical and practical issues.

This research project was titled as Chat message analysis based way-finding system along with social application and directory services. The system must be able to extract the given user user chat messages and provide the graphical view of his/her chat session. Chat message analysis application ,in my research area evaluate healthy is based on 3 main parts.

1. Extract fewer first – person pronouns
2. Fewer Exclusionary words.
3. Unusual details from the collected messages.

After going through above steps final outcome will be show to the user of the evaluate healthy of particular chat sessions.

2.1.1 Feasibility Study

There are many research papers and articles to get the main idea of the evaluate healthy of particular chat. Based way-finding and how to keep users interested in using the application. we have to make our own algorithm by enhance the existing weka algorithm. We have to ensure feasibility of that process in order to achieve target.

2.1.2 Requirement Analysis

After researching we found that currently existing systems are not a combination of real time chat message evaluation. Mostly the users are going to use separate applications for both things. And we found that there were lack of open source built applications and then we decide to use Open source technologies.

To gather information we used,

- Internet
- Related books
- Research Papers
- Human Views

2.1.3 System Analysis

As there are lots of applications based on chat message analysis team went through deep comparisons with most important applications like GroupWise, Analysis Messages , etc. Both Web and mobile applications were analyzed and there features and gray areas were captured.

2.1.4 System Design

The design part is done by dividing it into several parts.

- User interface designing: This will be the first part in designing because the user is interacting with the interface which is provided. So the interface must be user friendly and must fulfill all the needs and requirements of the user. And must be attractive to user's eye.
- Database design: For the database management we decide to use PostgreSQL[9] as it is more recommend for spatial data management. And System will consist with two databases spatial and non-spatial database. Database is being designed using logical database design concept and schema designing using Normalization.
- Algorithm design: We have enhanced existing algorithms for our use. They are Machine learning algorithms.

2.1.5 Implementation

According to the proposed system user have the facility to access our system with any device that has internet facility enabled. User should always on his chat session.

Whole system is developed using Open source software as it will be more accurate and useful in many ways.

- SQL was used in order to manage Databases.
- PHP was used as Server-side language.
- Java was used as developing algorithm and library’s.

Above used technologies were mainly used within the process and further classified below.

2.1.6 System Overview

2.2 Testing and Implementation

2.2.1 Implementation Techniques

- Servers required
  - Web Server - Appache Server
- To implement mobile web application
  - PHP
- To implement geo social mobile application
  - PHP
- PostgreSQL server for database designing
- Databases Implemented
  - Spatial Database to store Spatial information
  - Non-Spatial Database to Store other information
- Microsoft Office package for Documentation
- Hardware Requirements
  - Internet Connection
  - And device that has the facility to access internet
- Software Requirements
  - Web browser enables with internet access

2.2.2 Testing Techniques

- **Testing Methodology**

Any software product should be sent through a testing process in order to identify the weaknesses or faults of the system. This section provides an overview of how the system behaves in the phases recognized in the system. This is accomplished through the testing process where each and every module or phase of the system is tested.

- **Unit Testing**

Unit testing will be carried out during the implementation process. Each unit will be tested by the developer himself.

- **Socializing Concept Testing**

This was fully tested with all the team members and issues and attention needed points were identified. And few of the outsiders was given to use the application and got feed backs through them also as they go through the system they may do testing kind of thing.

- **Integration Testing**

Once more than one module is completed, integration testing will begin. Two modules will be integrated and tested. As a third module is joined, all three modules will be tested again.

- **System Testing**

Once all the modules have been successfully integrated, system testing will begin. Three kinds of System tests will be carried out. And for further testing application was given to some of the users and followed them.

2.2.3 User Interfaces

2.3 Research Findings

3 RESULT AND DISCUSSION

3.1 Results of the System

The following subsection provides evidence to the implementation results and the solutions provided for the identified research problems. The main user interface of the basic implementations of the "evaluate healthy of particular chat session "are shown with respective results followed by a small description of the displayed interface. So that all the interfaces were designed with a simple, attractive manner to keep the continuous user interaction.

3.1.1 Test Cases

3.2 Discussion of the System

As discussed earlier the main aim of coming up with this application to let analysis a chat message and get sprit of the total messages. So, by using our system we would provide the users more effective and efficient way evaluating chat session and to keep interact with user's friends.

The main system attributes are described as follows

- Reliability

When consider any system; System reliability is the most important part of it. Because, the entire system process is depending on that. If there is any crash or error occurs those can be recovered without any harm to the system or without any data losses. A system may come up with some hardware or software failures within its processing time. Hardware failures can be occurring more frequently than the software failures. To enhance that reliability in this system backup system will be used in order to ensure the system data security. It is very important to the user. Because if any case if the system crash user can rely on the backup. This system should be a highly reliable system, with a Mean Time to Failure greater than 8000 hours. The system should be tested for errors while developing units or modules. Then testing will be done

- Availability

When it comes to the availability it plays major role in this system. The system must be available for one user at a time for modifications. But many users chat messages evaluate at once.

- Security

Security one another important thing which must highly consider with the implementation of the system, the process of evaluating healthy of chat session depends on the knowledgebase. Therefore, we mainly concern about security requirement to protect the knowledgebase. We facilitate that system requirement by controlling the access to updating system. Only authorized people can access to that system by giving their username and password to do some changes. Unauthorized access to the database data is restricted.

- Maintainability

To provide more accurate responses to evaluate chat messages, we must maintain the knowledgebase. We can provide it by updating knowledgebase with most recent information. Programming shall be well commented and documented for any further development of the system. Moreover, must maintain the backup system too

3.2.1 Flow of the Project

4 CONCLUSION

REFERENCES

[1] Argamon, S., Saric, M., & Stein, S. S. (2003). Style mining of electronic messages for multiple authorship discrimination: first results. In Proceedings of the ninth ACM

SIGKDD international conference on Knowledge discovery and data mining (pp. 475–480). Washington, D.C., USA.

[2] Baayen, R. H., van Halteren, H., & Tweedie, F. J. (1996). Outside the cave of shadows: using syntactic annotation to enhance authorship attribution. Literary and Linguistic Computing, 11(3), 121–132.

[3] Backer, E., & Kranenburg, P van. (2004). Musical style recognition - a quantitative approach. In Proceedings of the Conference on Interdisciplinary Musicology (CIM04) Graz, Austria.

[4] Binongo J. N. G., & Smith M. W. A. (1999). The application of principal component analysis to stylometry. Literary and Linguistic Computing, 11(3), 121–131.

[5] Burrows, J. (1987). Computation into criticism: A study of Jane Austen’s novels and an experiment in method. Oxford: Clarendon Press.

[6] Kessler, B., Nunberg, G., & Schutze, H. (1997). Automatic detection of text genre. In Proceedings of the 35th Annual Meeting on Association for Computational Linguistics (pp. 32–38). Madrid, Spain.

APPENDICES

Appendix A: Source Code

```
public class TextClassifier {

        Connection conn = null;
    com.mysql.jdbc.PreparedStatement pst = null;

    private String[]   inputText       = null;
    private String[]   inputClasses    = null;
    private String     classString     = null;

    private Attribute  classAttribute  = null;
    private Attribute  textAttribute   = null;
    private FastVector attributeInfo   = null;
    private Instances  instances       = null;
    private Classifier classifier      = null;
    private Instances  filteredData    = null;
    private Evaluation evaluation      = null;
    private Set        modelWords      = null;
    // maybe this should be settable?
    private String     delimitersStringToWordVector = "\\s.,:'\\\"()?!";

    //
    //
    //
    public static void main(String args[]) {

        String classString = "weka.classifiers.bayes.NaiveBayes";
        String thisClassString = "weka.classifiers.lazy.IBk";

        if (args.length > 0) {
            thisClassString = args[0];
        }
      // String[] inputText =  {"you want to buy from me?",};



       String message="";
       for(String token:args){

       message+=token+" ";

       }

HashSet classSet = new HashSet(Arrays.asList(inputClasses));
        classSet.add("?");
        String[] classValues = (String[])classSet.toArray(new String[0]);

        //
        // create class attribute
        //
        FastVector classAttributeVector = new FastVector();
        for (int i = 0; i < classValues.length; i++) {
            classAttributeVector.addElement(classValues[i]);
        }
        Attribute thisClassAttribute = new Attribute("class", classAttributeVector);
        //
        // create text attribute
        //
        FastVector inputTextVector = null;  // null -> String type
        Attribute thisTextAttribute = new Attribute("text", inputTextVector);
        for (int i = 0; i < inputText.length; i++) {
           thisTextAttribute.addStringValue(inputText[i]);
        }


        for (int i = 0; i < testText.length; i++) {
           thisTextAttribute.addStringValue(testText[i]);


        }



        FastVector thisAttributeInfo = new FastVector(2);
        thisAttributeInfo.addElement(thisTextAttribute);
        thisAttributeInfo.addElement(thisClassAttribute);




        TextClassifier classifier = new TextClassifier(inputText, inputClasses, thisAttributeInfo,
thisTextAttribute, thisClassAttribute, thisClassString);
//edit
     //  System.out.println("DATA SET:\n");



        System.out.println(classifier.classify(thisClassString));


        //System.out.println("NEW CASES:\n");
    //  System.out.println(classifier.classifyNewCases(testText));



    }







    TextClassifier(String[] inputText, String[] inputClasses, FastVector attributeInfo, Attribute
textAttribute, Attribute classAttribute, String classString) {
        this.inputText      = inputText;
        this.inputClasses   = inputClasses;
        this.classString    = classString;
        this.attributeInfo  = attributeInfo;
        this.textAttribute  = textAttribute;
        this.classAttribute = classAttribute;


    }

    TextClassifier() {
      //  throw new UnsupportedOperationException("Not supported yet."); //To change body
of generated methods, choose Tools | Templates.
    }
    public StringBuffer classify() {

        if (classString == null || "".equals(classString)) {
           return(new StringBuffer());

        }

        return classify(classString);


    }
    public StringBuffer classify(String classString) {

        this.classString = classString;

        StringBuffer result = new StringBuffer();

        // creates an empty instances set
        instances = new Instances("data set", attributeInfo, 100);

        // set which attribute is the class attribute
        instances.setClass(classAttribute);


        try {

            instances = populateInstances(inputText, inputClasses, instances, classAttribute,
textAttribute);
         //   result.append("DATA SET:\n" + instances + "\n");


            // make filtered SparseData
            filteredData = filterText(instances);

            // create Set of modelWords
            modelWords = new HashSet();
            Enumeration enumx = filteredData.enumerateAttributes();
            while (enumx.hasMoreElements()) {
                Attribute att = (Attribute)enumx.nextElement();
                String attName = att.name().toLowerCase();
                modelWords.add(attName);



            }
            // Classify and evaluate data
            //
            classifier = Classifier.forName(classString,null);


            classifier.buildClassifier(filteredData);
            evaluation = new Evaluation(filteredData);


           evaluation.evaluateModel(classifier, filteredData);


// Print the instance

           // evaluation.evaluateModelOnceAndRecordPrediction(evaluat, null);




          result.append(printClassifierAndEvaluation(classifier, evaluation) + "\n");
           //  check instances
       //    int startIx = 0;
     //      result.append(checkCases(filteredData, classifier, classAttribute, inputText, "not
test", startIx)  + "\n");


        } catch (Exception e) {
            e.printStackTrace();
            result.append("\nException (sorry!):\n" + e.toString());
        }

        return result;

    }
    public StringBuffer classifyNewCases(String[] tests) {

        StringBuffer result = new StringBuffer();


        Instances testCases = new Instances(instances);
        testCases.setClass(classAttribute);



        String[] testsWithModelWords = new String[tests.length];
        int gotModelWords = 0; // how many words will we use?

        for (int i = 0; i < tests.length; i++) {
            // the test string to use
            StringBuffer acceptedWordsThisLine = new StringBuffer();

            // split each line in the test array
            String[] splittedText = tests[i].split("["+delimitersStringToWordVector+"]");
            // check if word is a model word
            for (int wordIx = 0; wordIx < splittedText.length; wordIx++) {
                String sWord = splittedText[wordIx];
                if (modelWords.contains((String)sWord)) {
```

```
                gotModelWords++;



                acceptedWordsThisLine.append(sWord + " ");
            }
        }
        testsWithModelWords[i] = acceptedWordsThisLine.toString();


    }


    // should we do do something if there is no modelWords?
    if (gotModelWords == 0) {
        result.append("\nWarning!\nThe text to classify didn't contain a single\nword from
the modelled words. This makes it hard for the classifier to\ndo something usefull.\nThe
result may be weird.\n\n");
    }

    try {

        // add the ? class for all test cases
        String[] tmpClassValues = new String[tests.length];
        for (int i = 0; i < tmpClassValues.length; i++) {
            tmpClassValues[i] = "?";
        }

        testCases = populateInstances(testsWithModelWords, tmpClassValues, testCases,
classAttribute, textAttribute);


        // result.append("TEST CASES before filter:\n" + testCases + "\n");

        Instances filteredTests = filterText(testCases);


        int startIx = instances.numInstances();
        result.append(checkCases(filteredTests, classifier, classAttribute, tests, "newcase",
startIx) + "\n");

    } catch (Exception e) {
        e.printStackTrace();
        result.append("\nException (sorry!):\n" + e.toString());
    }

    return result;

  } //  end classifyNewCases


  //
  //  from empty instances populate with text and class arrays
  //
  public static Instances populateInstances(String[] theseInputTexts, String[]
theseInputClasses, Instances theseInstances, Attribute classAttribute, Attribute textAttribute)
{
        for (int i = 0; i < theseInputTexts.length; i++) {
           Instance inst = new Instance(2);
           inst.setValue(textAttribute,theseInputTexts[i]);
           if (theseInputClasses != null && theseInputClasses.length > 0) {
              inst.setValue(classAttribute, theseInputClasses[i]);
           }
           theseInstances.add(inst);
        }

       return theseInstances;


    } // populateInstances





    //result all the instances

    //
    // check instances (full set or just test cases)
    //
    public static StringBuffer checkCases(Instances theseInstances, Classifier thisClassifier,
Attribute thisClassAttribute, String[] texts, String testType, int startIx) {

        StringBuffer result = new StringBuffer();


        try {

         //   result.append("\nCHECKING ALL THE INSTANCES:\n");

           Enumeration enumClasses = thisClassAttribute.enumerateValues();
        //   result.append("Class values (in order): ");
           while (enumClasses.hasMoreElements()) {
              String classStr = (String)enumClasses.nextElement();
          //     result.append("'" + classStr + "'  ");
              }
         //  result.append("\n");

           // startIx is a fix for handling text cases
           for (int i = startIx; i < theseInstances.numInstances(); i++) {

              SparseInstance sparseInst = new SparseInstance(theseInstances.instance(i));
              sparseInst.setDataset(theseInstances);

          //     result.append("\nTesting: '" + texts[i-startIx] + "'\n");

              // result.append("SparseInst: " + sparseInst + "\n");

              double correctValue = (double)sparseInst.classValue();
              double predictedValue = thisClassifier.classifyInstance(sparseInst);

              String predictString = thisClassAttribute.value((int)predictedValue) + " (" +
predictedValue + ")";
```

```
            //      result.append("predicted: '" + predictString);

            // print comparison if not new case
            if (!"newcase".equals(testType)) {
                String correctString = thisClassAttribute.value((int)correctValue) + " (" +
correctValue + ")";
                String testString = ((predictedValue == correctValue) ? "OK!" : "NOT OK!") +
"!";


        //         result.append("' real class: '" + correctString +  "' ==> " +  testString);
            }
          //   result.append("\n");



          //   result.append("\n");

          }

      } catch (Exception e) {
          e.printStackTrace();
          result.append("\nException (sorry!):\n" + e.toString());
      }

      return result;

  }




  public static Instances filterText(Instances theseInstances) {

      StringToWordVector filter = null;
      // default values according to Java Doc:
      int wordsToKeep = 1000;

      Instances filtered = null;

      try {

          filter = new StringToWordVector(wordsToKeep);
          // we ignore this for now...
          // filter.setDelimiters(delimitersStringToWordVector);
          filter.setOutputWordCounts(true);
          filter.setSelectedRange("1");

          filter.setInputFormat(theseInstances);

          filtered = weka.filters.Filter.useFilter(theseInstances,filter);
          // System.out.println("filtered:\n" + filtered);

      } catch (Exception e) {
        e.printStackTrace();
      }

      return filtered;

   } // end filterText


   //
   // information about classifier and evaluation
   //
   public static StringBuffer printClassifierAndEvaluation(Classifier thisClassifier,
Evaluation thisEvaluation) {

      StringBuffer result = new StringBuffer();

      try {

   result.append("\n\nINFORMATION ABOUT THE CLASSIFIER AND
EVALUATION:\n");

  result.append("\nclassifier.toString():\n" + thisClassifier.toString() + "\n");


    result.append("\nevaluation.toSummaryString(title, false):\n" +
thisEvaluation.toSummaryString("Summary",false)  + "\n");

         String[] split = thisEvaluation.toSummaryString("Summary",false).split(" ");

        //  StringBuffer secondPart = new StringBuffer();
         // System.out.println(""+split.length);

         /* for (int i = 1; i < split.length; i++) {
             if(!split[i].isEmpty()){
                  if(i==12)
                  result.append(split[i]+"");
             }
         }

                */
   result.append("\nevaluation.toMatrixString():\n" + thisEvaluation.toMatrixString()  + "\n");

   result.append("\nevaluation.toClassDetailsString():\n" +
thisEvaluation.toClassDetailsString("Details")  + "\n");

      result.append("\nevaluation.toCumulativeMarginDistribution:\n" +
thisEvaluation.toCumulativeMarginDistributionString()  + "\n");






      } catch (Exception e) {
         e.printStackTrace();
```

```
            result.append("\nException (sorry!):\n" + e.toString());
        }

        return result;

    } // end printClassifierAndEvaluation



    //
    // setter for the classifier _string_
    //
    public void setClassifierString(String classString) {
        this.classString = classString;
    }

    static class trainSentDectectModel {

        public trainSentDectectModel() {
        }
    }


}
```

Appendix B: Class Diagram