Automated Rule Extraction Over A Scientific Text Data Warehouse Using A Domain-Specific Markup Language

Master's Project submitted to
The Department of Computer Science at Montclair State University

In Partial fulfillment of the requirements for a degree in Master of Science in Computer Science

Muharrem Aker
Faculty Advisor: Prof. Aparna Varde
December 2009

This research is in collaboration with the Center for Heat Treating Excellence Material Science and Engineering Worcester Polytechnic Institute, Worcester, MA

TABLE OF CONTENTS

ABSTRACT

In this work, we propose a technique called RuleExtractor to automatically derive association rules from plain text sources in a scientific data warehouse using a domain-specific markup language. More specifically, QuenchML a markup language for the heat treating materials is used to convert plain text to structured text. This helps to perform rule discovery using the Apriori algorithm for Association Rule Mining. However, RuleExtractor is not limited to QuenchML and it can potentially be extended to other markup languages. The rules are applicable in the real world and can be used as input to various expert systems in order to avoid the lengthy detailed discussions associated with having domain experts manually infer the rules.

1. INTRODUCTION

1.1 Background

An Expert System is a computer application that performs a task that would otherwise be performed by a human expert [21]. Some of the rules that operate the system are often pulled out from the related text sources in the literature. In mining rules there should be human judgment that incorporates the domain knowledge. The system we developed automates the human judgment in scientific text mining. We utilize a previously developed XML based standard in our system. XML (Extensible Markup Language) is a widely accepted standard for storing and transporting data on the web. It is also used as a descriptive framework for domain specific markup languages [21]. A domain specific markup language has a certain set of tags that holds the semantics and structures of a domain. An example is the QuenchML that has been developed in Worcester Polytechnic Institute. QuenchML is a domain specific markup language that aims to serve as a standard in the area of Heat Treating of Materials. Quenching, i.e., the rapid cooling of materials forms an important step in the heat treating operations [2]. It is composed of XML tags that are specific to Heat Treating of Materials. These are useful to mark up text archived in a customized repository, i.e., data warehouse. The term data warehouse was first defined by Bill Inmon in 1990 which he described as a "subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process." [20]. For this purpose, relevant literature in the heat Heat Treating domain is gathered together, in a customized repository in the form of text source of information.

1.2 Problem Definition

We are interested in developing a tool to automatically derive association rules from a scientific data warehouse using a markup language. Specifics of the RuleExtractor are described below in detail.

Input: The input to RuleExtractor is the literature on the subject of a scientific domain. Here, we focus on the domain "Heat Treating of Materials". For this purpose, academic papers have been gathered during the project and stored in plain text format in a customized repository that serves as a data warehouse in this scientific domain.

Goal: The goal of this work is to obtain association rules from the input text data stored in a data warehouse.

Process: The association rules that represent the knowledge of domain are to be obtained via use of our RuleExtractor approach which combines natural language processing with association rule mining, using the domain specific markup language as a guideline.

Output: Consists of association rules that represent domain knowledge.

2. PROPOSED APPROACH: RULEEXTRACTOR

The system architecture of our RuleExtractor approach is given in Figure 1. PDF files are obtained from Center for Heat Treating Excellence (CHTE) on the subject of Heat Treating of Materials, with emphasis on Quenching or the rapid cooling step.

In our approach, we convert these files into .txt format and store them in a data warehouse. In order to proceed with extraction, we convert the plain text data into structured data. For this, we deploy a natural language parser in our system, i.e., a program that works out the grammatical structure of sentences, for instance, which groups of words go together (as "phrases") and determine which words are the subject or object of a verb [8]. More specifically, we utilize the Stanford Parser, an open source natural processing tool written in Java. In the next step of our rule extraction, we use the classic algorithm called Apriori to derive association rules from the structured data. To achieve this, we incorporate the well known data mining tool WEKA (Waikato Environment for Knowledge Analysis) in our system architecture.

Below we describe each component of the RuleExtractor in more detail.

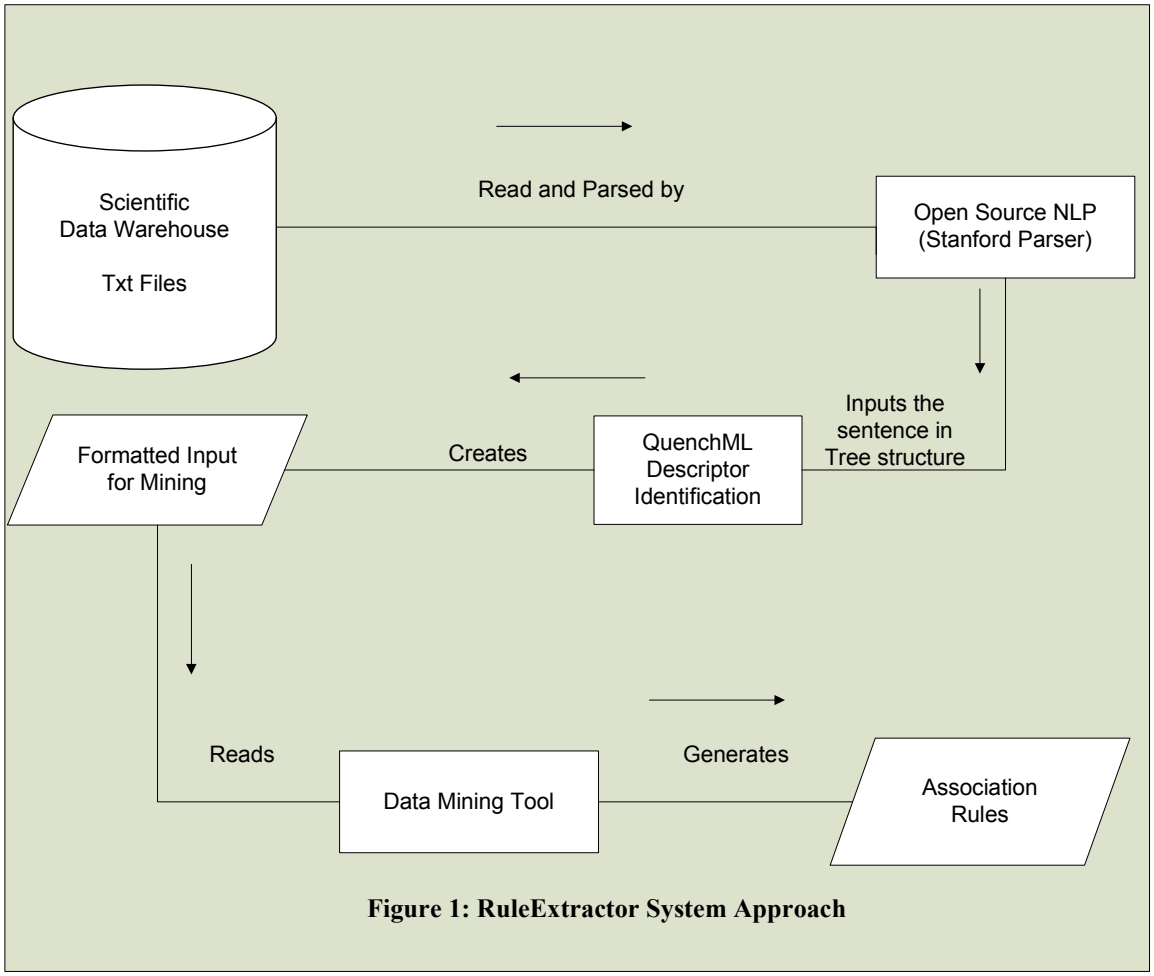


Figure 1: RuleExtractor System Approach

2.1 Obtaining Structured Data

The Stanford parser reads the text file sentence by sentence and gives the result in a syntactic structure. First we need to analyze the tree output by Stanford parser. Each word in a sentence has a specific tag. Each tag is located on a node of the tree and all the other words composing the plain text data will be located in the leaves. Since our focus is to primarily identify QuenchML tags and their descriptors, the trees generated by the parser are searched for those tags.

The Stanford parser deploys Treebank tags that have been developed by The University of Pennsylvania. Essentially, Treebank is a corpus of parsed sentences used by many researchers for training data-driven parsing algorithms [7]. Some of more common tags that are also relevant to our task are: simple declarative clause (S), noun phrase (NP), noun (NN), coordinating conjunction (CC), and adjective (JJ). Further information on other Treebank tags and their descriptions can be found at http://bulba.sdsu.edu/jeanette/thesis/PennTags.html. Figure 2 exemplifies a tree structure

governing the grammatical structure of the sample text "**Moderate agitation and thick surface area implies low distortion tendency**" as utilized by the Stanford parser. It can be seen in Figure 2 that the node contains the grammatical composition of the sentence and the leaves have the words.

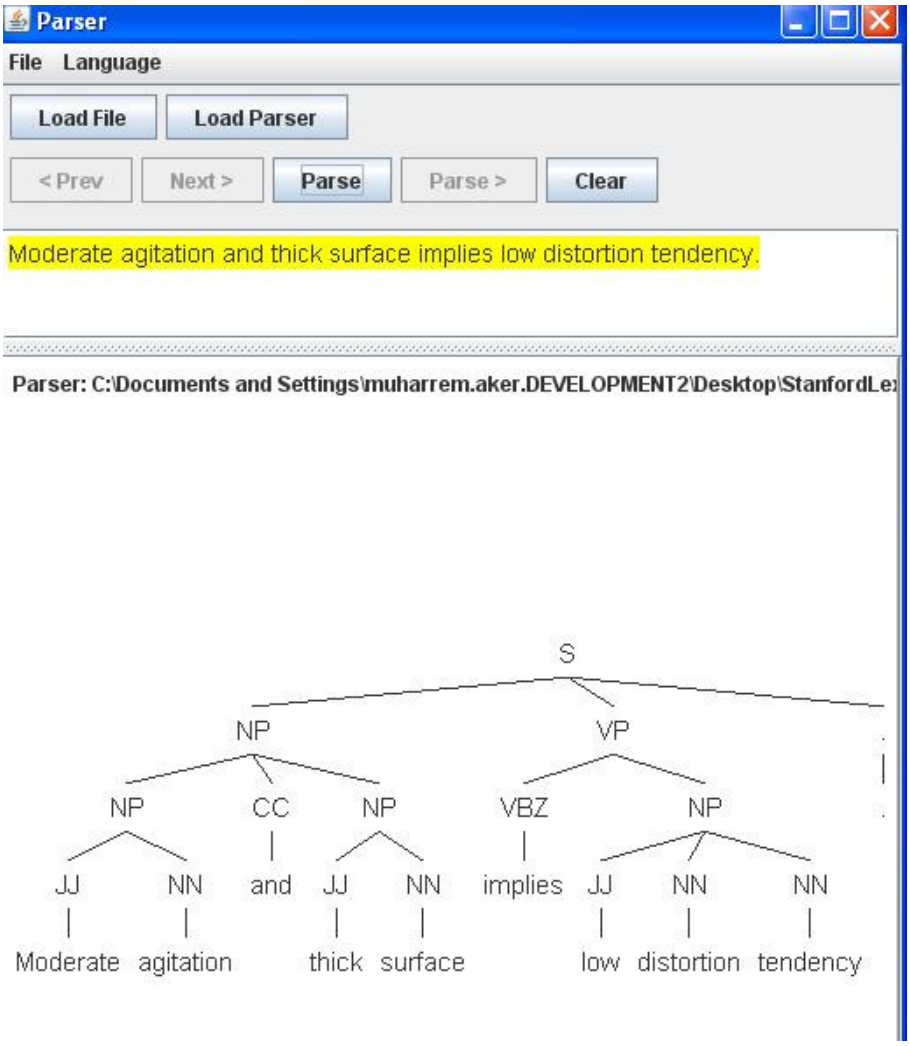


Figure 2: Tree structure of an English sentence

Below is the actual output of the parser described in Figure 2 that is used by our code to identify all the QuenchML tags and their descriptors (when they exist).

(S
  (NP
    (NP (JJ **Moderate**) (NN **agitation**))
    (CC **and**)
    (NP (JJ **thick**) (NN **surface**) (NN **area**)))
  (VP (VBZ **implies**)

(NP (JJ **low**) (NN **distortion**) (NN **tendency**))))

From the above sample output (corresponding to Figure 2) it can be observed that the words like agitation, distortion and surface are parsed as NN (Noun) and their descriptive words are always parsed as JJ (Adjective) tags. It can also be clearly seen from Figure 2 that JJ and NN tags are the children of NP's (Noun Phrase).

Given the above structured text our system checks if there is any predefined QuenchML tags, like 'agitation', 'surface' and 'distortion', in the structured data and determines whether those tags are accompanied with any descriptive words. This process continues until the end of the text file is reached.

Overall, our algorithm can be reduced to the following three inferences, given the parsed plain text data. These are:

1- Find if NP exists
2- Find if NN exists under NP.
3- Find if JJ exists corresponds to NP that is found on step 2.

The structured data generated from the plain text data after parsing and these steps are then statistically analyzed to extract associate rules.

Finally, after completing these steps, above given plain text will be converted into

structured text as follows:

<Distortion> Low **</Distortion >**

<Agitation > Moderate **</Agitation>**

<Surface > Thick **</Surface>**

In the above structured text, 'distortion', 'agitation' and 'surface' are the tags specific to heat treatment of materials domain and 'low', 'moderate', and 'thick' are the descriptive words for those QuenchML tags.

2.2 Data Mining

The structured data is next subjected to data mining to ultimately obtain the association rules. Essentially, in data mining, association rule looks for interesting relationships among items in a given data set.

Association Rule Mining is well known and an adapted method for discovering interesting relations between variables in a repository. The analysis focuses on strong rules discovered in databases using different measures of interestingness [7]. Association rule mining can be explained with a concept of "market basket analysis" which identifies customers purchasing behaviors. It tries to find out a relation between the purchased products. The ultimate goal is to determine best rules that govern the purchase of sequence of product combinations, the time of purchase, and in what sequence items were purchased. The resulting relation data is then recorded and stored in a repository for further manipulation and inference. Not only do these findings enable a business to increase their sales by promotional pricing or product placements but also help customers to buy items that might have been forgotten otherwise.

The interestingness of a rule can be measured by the concepts of rule support and rule confidence. The confidence and support of a rule are defined as follows.

The support of an association rule is the percentage of the population which satisfies the rule.

$A \Rightarrow B$

Support $(A \Rightarrow B) = P(A \cap B)$

$$\text{Support } (A \Rightarrow B) = \frac{\text{Number of tuples containing both A and B}}{\text{Total number of tuples}}$$

On the other hand confidence is defined as the measure of certainty or reliability associated with each determined pattern.

$A \Rightarrow B$

Confidence $(A \Rightarrow B) = P(B|A)$ (The probability of B given that all we know is A)

A rule is considered to be interesting if it satisfies the minimum threshold and minimum confidence thresholds. In other words, rules are derived for predetermined confidence and support levels, and all rules that are below the designated support and confidence levels are discarded.

These rules derived using the Apriori algorithm for mining frequent item sets was introduced by Agrawal et al [22] for discovering regularities between products in a transaction.

This algorithm is based on the use of "prior knowledge" of associated items. In our work, to obtain the support and confidence values in the given structured data after conversion, each sentence is examined for dependencies between QuenchML tags and their descriptors.

In our implementation, after all tags and their descriptors are identified they are converted into the .arff format as the input to the data mining tool WEKA. WEKA is used to implement Apriori algorithm to obtain association rules.

Weka supports four different data types in version 3.2.1. [24]

* numeric
* nominal-specification
* string
* date

We created the .arff file in nominal-specification. RuleExtractor formats the result similar to the example below. A sample WEKA file consists of the following, in our context.

@relation heattransfer.symbolic
@attribute Agitation {moderate, Low, High}
@attribute Distortion {low,High }
@attribute Surface {thick,Thin }
@data
Moderate, low, thick,

Likewise, there are other attribute pertaining to various properties of the data. Note that the descriptor word in {} are for the given example. We can have other descriptors.

2.3 Implementation

Input to this code is a text file and the output is .arff file which is generated by RuleExtractor using the approach in Figure 1. This serves as an input to WEKA tool in our system to derive association rules. The code developed to implement in this project

shown in Appendix A. In our experiment, we utilized Dell Vostro 1510 laptop with the configuration of Windows Vista business SP1 operating system (32 bit), Intel Core 2 Duo CPU, 3 GB of Memory and 250 GB of hard disk. We implemented our application with Netbeans 6.01 IDE and Weka 3.6.1.

3. EXPERIMENTAL EVALUATION

The experimental results are summarized below.

3.1Manual Approach

Method:

Manual Approach involved following steps:

- We found a few domain related literature sources on the web and save them in a txt format.
- Then we manually searched and found the QuenchML tags and their corresponding descriptors.
- All the keywords and descriptors were saved in special text files (.arff format).
- Utilized WEKA to see the result by opening .arff file. WEKA takes .arff file as an input and produces the rule set after analyzing the attributes and values.

After the steps mentioned above processed, we found the result below

SAMPLE OF BEST RULES FOUND

1. Cracking=Likely_to_Occur == Part=Residual_Stress

2. Part=Residual_Stress == Cracking=Likely_to_Occur

3. CrossSection=Thin == Cracking=Likely_to_Occur Degradation=No Fixture=Improper GrainNature=Non_Uniform Impellers=Not_Used Orientation=Transverse Part=Residual_Stress SpeedImprovers=Not_Used Stamping=Not_Occured Warpage=Thick Welding=Not_Used

4. Degradation=No == Cracking=Likely_to_Occur CrossSection=Thin Fixture=Improper GrainNature=Non_Uniform Impellers=Not_Used

Orientation=Transverse Part=Residual_Stress SpeedImprovers=Not_Used Stamping=Not_Occured Warpage=Thick Welding=Not_Used

5. Fixture=Improper == Cracking=Likely_to_Occur CrossSection=Thin Degradation=No GrainNature=Non_Uniform Impellers=Not_Used Orientation=Transverse Part=Residual_Stress SpeedImprovers=Not_Used Stamping=Not_Occured Warpage=Thick Welding=Not_Used

6. GrainNature=Non_Uniform == Cracking=Likely_to_Occur CrossSection=Thin Degradation=No Fixture=Improper Impellers=Not_Used Orientation=Transverse Part=Residual_Stress SpeedImprovers=Not_Used Stamping=Not_Occured Warpage=Thick Welding=Not_Used

7. Impellers=Not_Used == Cracking=Likely_to_Occur CrossSection=Thin Degradation=No Fixture=Improper GrainNature=Non_Uniform Orientation=Transverse Part=Residual_Stress SpeedImprovers=Not_Used Stamping=Not_Occured Warpage=Thick Welding=Not_Used

8. Orientation=Transverse Cracking=Likely_to_Occur CrossSection=Thin Degradation=No Fixture=Improper GrainNature=Non_Uniform Impellers=Not_Used Part=Residual_Stress SpeedImprovers=Not_Used Stamping=Not_Occured Warpage=Thick Welding=Not_Used

9. SpeedImprovers=Not_Used == Cracking=Likely_to_Occur CrossSection=Thin Degradation=No Fixture=Improper GrainNature=Non_Uniform Impellers=Not_Used Orientation=Transverse Part=Residual_Stress Stamping=Not_Occured Warpage=Thick Welding=Not_Used

10. Stamping=Not_Occured == Cracking=Likely_to_Occur CrossSection=Thin Degradation=No Fixture=Improper GrainNature=Non_Uniform Impellers=Not_Used Orientation=Transverse Part=Residual_Stress SpeedImprovers=Not_Used Warpage=Thick Welding=Not_Used

3.2 Automated Approach

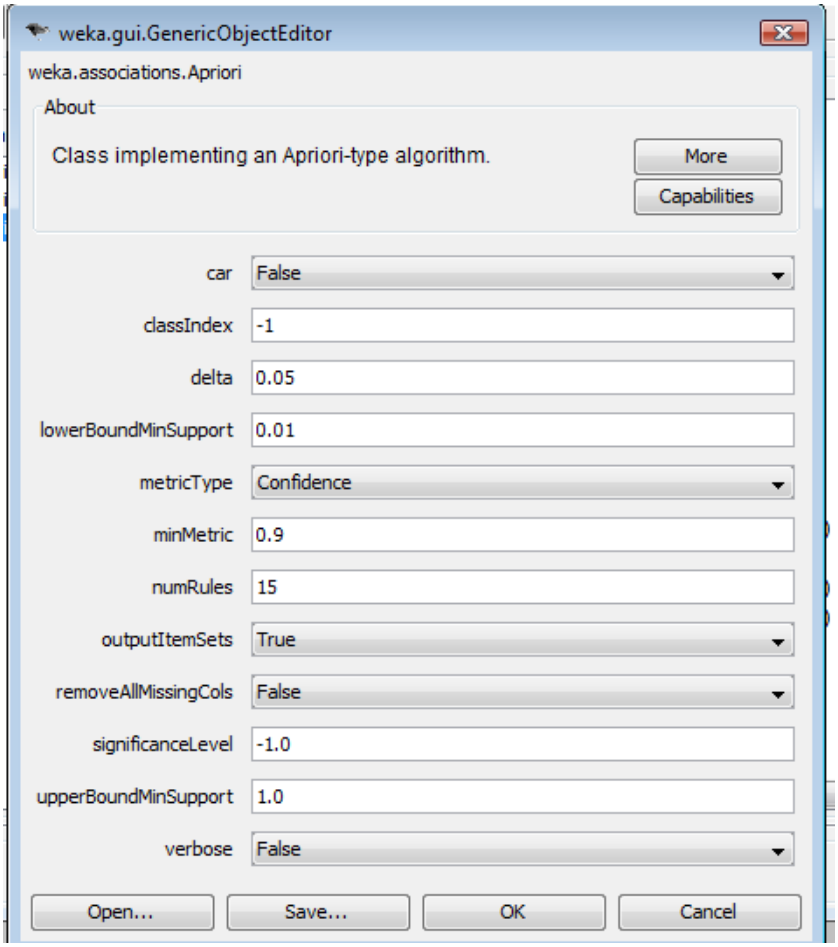
Method:
Automated approach involved following steps:

- We saved the obtained literature in a repository
- We downloaded a java IDE Netbeans 6.0. We added the implementation of RuleExtractor to Stanford parser.
- We then ran the new system and created an .arff file to hold the attribute and instance data.

- ❖ We opened the .arff file with WEKA and applied the Association Rule Mining algorithm Apriori along with sparse to non sparse filters. WEKA created the result set.

After the steps mentioned above processed, we found the result below.

WEKA settings before run.



=== Run information ===

Scheme: weka.associations.Apriori -I -N 15 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.01 -S -1.0 -c -1
Relation: heattransfer.symbolic
Instances: 126
Attributes: 19

SAMPLE OF BEST RULES FOUND (1)

1. HeatTransferCoefficient=high ==> Agitation=high
2. Agitation=high ==> HeatTransferCoefficient=high
3. Distortion=more ==> Agitation=higher
4. Agitation=higher ==> Distortion=more
5. Geometry=different ==> Quench=different

6. Quenchant=same ==> Quench=different
7. Quench=portable ==> Quenchant=different
8. Quenchant=different ==> Quench=portable
9. Surface=rapid ==> Quench=intensive
10. Type=certain ==> QuenchSeverity=molten
11. QuenchSeverity=molten ==> Type=certain

The .arff file for the above result set can be found in Appendix B.

SAMPLE OF BEST RULES FOUND (2)

1. HeatTransferCoefficient=high == Agitation=high
2. Agitation=high == HeatTransferCoefficient=high
3. Distortion=more == Agitation=higher
4. Agitation=higher == Distortion=more
5. Geometry=different ==Quench=different
6. Quenchant=same == Quench=different
7. Quench=portable == Quenchant=different
8. Quenchant=different == Quench=portable
9. Surface=rapid == Quench=intensive
10. Type=certain == QuenchSeverity=molten

Note that we used WEKA filters such as Attribute Selection to further preprocess the data for Association Rule Mining. We considered multiple runs of the system. We got several interesting rules, among which a few samples are shown here.

4. RELATED WORK

There is considerable interest in the field of Text Mining. Most research in this field focuses on mining in order to improve text retrieval. The goal in most cases is to have more relevant searches of documents from repositories like the Web and others. The work in [27] introduces a preprocessor that parses web pages into logically coherent segments, and an NLP system that learns text extraction rules from example. In [15] an approach is suggested for generating association rules that relates structured data values to concepts extracted from unstructured data, using an Extended Concept Hierarchy that focuses mainly on parent-child and sibling relationships between concepts. Thomere et al in [14] describe an ontology system that is built with the goal of enabling domain experts to build knowledge bases without relying on AI scientists and engineers. In the work

described in [10] domain-specific dictionaries are created for information extraction given an appropriate training corpus. The tool Wordnet [9] is a lexical database that attaches different possible meanings to words in several contexts. IBM's Intelligent Miner for Text [11] offers a wide range of tools for text analysis.

Our technique RuleExtractor derives rules by converting the plain text to structured text, aiming to automate the rule discovery otherwise performed with the help of human experts. Moreover, RuleExtractor emphasizes the application of a user defined markup language, within the context of XML, a lingua franca for information exchange today.

5. CONCLUSION

Throughout this project, I used variety of environments such as Netbeans IDE, Stanford parser, QuenchML and more importantly Weka data mining software. There are many steps involved in this project.

1. At least 100 articles found in regards to Heat Treating Materials and saved them in a Data Warehouse in txt format.
2. Read this file sentence by sentence with java code that I implemented which couples with Stanford parser. The program finds the tag(s) which previously determined by QuenchML, in the sentence along with its descriptor(s) if there exists.
   In other words our program reads the sentence from the data warehouse and compares against the QuenchML tags which are stored in the program. If the tag(s) that we look for in the given sentence are found the program goes to the next step and searches the descriptive words which are associated with the tag(s). If the tag(s) and descriptor(s) found together then the program saves the result as an attribute instance relation in the memory for later output. If the tag(s) found and there was no descriptive word then the program disregards that sentence and goes to the next sentence. This is a continuing process until the end of file mark obtained from the Data Warehouse. Therefore in this project, the sentences are read by the program is in fact important for the success of the project.
   For example, Moderate agitation and thick surface area implies low distortion tendency, is a good sentence for the project. The program parses the sentence and converts it to QuenchML.

   <Agitation> Moderate</ Agitation >
   < Surface > Thick </ Surface >
   <Distortion> Low </Distortion>

On the other hand the sentence, given the input conditions of a quenching experiment,
estimate the resulting heat transfer curve that would be obtained, contains our tags Quenching and Heat Transfer Curves is not a good sentence for the program since it does not have any descriptive word in the sentence.

3. After reading all sentences, found tags and their descriptors outputted to text file named WekaResult.arff in a specific formatting that are accepted by WEKA software. This step involves many conversions and formatting. The program I wrote accomplishes this tag as well. The .arff format requires below :

    a) .ARFF file variable declarations requires:
    @relation +Relation Name. Symbolic
    Example: relation heattransfer.symbolic
    b) Each attribute starts with "@attribute " marks and requires new line.
    c) Attribute instance format requires:
    @attribute +attributename+ {instances1, instances 2, instancesn}
    Example: @attribute HeatTransferCoefficient {artificial, vacuum, standard}
    Weka does not allow space in the attribute name. Our program finds the attribute for instance "Heat transfer Coefficient" and removes the white space before saving in the .arff file format. For the instances there is a requirement as well. If program finds same instance more than once for the same attribute then redundant instances need to be removed from the instance list. These requirements are considered and taken care of automatically.
    d) Arff data instances starts as:
    @data
    e) Each sentence corresponds one line and contains the descriptor that matches with an attribute.
    Example: Attribute1 (instance(x)), Attribute2 (instance), Attribute n (instance)
    There is semicolon between each instance and after the last instance no semicolon. Furthermore if the instance does not exist in the sentence then question mark needs to be inserted.
    Example 2:
    ?, ?, ?, ?, variable, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, extended, ?

    After saving everything in the .arff format we run the Weka apriori algorithm to find the association rules.

I contributed this project by developing a java application called RuleExtractor which accomplishes the tasks explained above. In formality, the application discovers

Association rules from given plain text using a markup language. RuleExtractor focuses on extracting rules from plain text data in Heat Treating Materials and uses QuenchML tags at its core, in our implementation.

Our program deploys natural language processing procedures to obtain structured data from the input plain text data. The resulting structured data is then analyzed utilizing Apriori algorithm to obtain rules with predetermined minimum support and confidence levels.

Our implementation couples open source software with JAVA code we developed. The results obtained with both real and plain text data show that association rules can be successfully derived.

As expected, rules get more specific as the occurrence frequency of QuenchML tags in the underlying text increases. Furthermore it is seen that the less sparse the instances the better the result obtained. Finding better rules with RuleExtractor requires that a sentence should hold at least two or more QuenchML tags. The rules derived are useful in real world applications such as decision support systems.

6. ACKNOWLEDGMENTS

I would like to acknowledge and extend my heartfelt gratitude to my project advisor Dr. Aparna Varde, for her vital encouragement and support. I got proper guidance at every step of this project due to the time and effort she put throughout the project and the sources that she provided. All the materials, books, web sites that she provided were very appropriate and helpful for my project.

I would also like to thank Prof. Richard Sisson, Director of Material Science Program and Dean of Graduate Studies at Worcester Polytechnic Institute (WPI). His valuable assistance in the collection of the literature in the subject of Heat Treating of Materials was extremely useful. I would like to also thank the faculty staff and students in Department of Computer Science at Montclair State University for their co-operation.

APPENDIX A:

PROGRAM IMPLEMENTATION

```
class XmlBuilder{

public boolean    hasRequiredTags; //If my tags in the sentence
public String[] tempValues;//Temporary Data instance location
public static String[] myTags = {
"Age","Agitation","Alloy","Bowing","Carbon Content","Cold Plastic
Formation", "Cooling Curve","Cooling Rate","Cooling Uniformity",
"cracking","Cross section",
"Degradation","Distortion","Fixture","Geometry ","Quenchometer",
"Grain Nature","Hardness","Heat Transfer Coefficient",
"Impellers","Orientation","Oxide","Part","Quenchant",
"Quench","Quenching","Quench Severity","Residual Stress","Size","Speed
"Improvers","Stamping", "Surface","Temperature","Type","Viscosity",
"Warpage","Welding"};

 public static ArrayList[] listArray;//Array of adjectives
 public static StringBuilder dataResult;//weka data instance

 public static boolean[] activeTags;// Active tags. check whether the
//tags are used to filter the unused tags
```

Initilaze the Static variables that are defined previously.

```
 public static void InitStaticMembers(){

        listArray = new ArrayList[myTags.length];
        activeTags = new boolean[myTags.length];

        for( int i=0; i<listArray.length; i++){
            listArray[i]  = new ArrayList();
            activeTags[i] = true;
        }
        dataResult = new StringBuilder();
    }

  //Constructor  fill temp values with question mark for Weka data
//instances

public XmlBuilder(){
        this.hasRequiredTags = false;
        this.tempValues = new String[ myTags.length ];
        for(int i=0; i<this.tempValues.length; i++){
            this.tempValues[i] = "?";
        }
    }

    public String traverseTree(Tree node){

     String labelNode = node.label().toString();
     Tree [] trChildren = node.children();//array of chldrens as a tree
        String[] strLabels = new String[node.children().length];
        ArrayList listNN = new
ArrayList();//["heat","transfer","coefficent"] .
```

```
        ArrayList listJJ = new ArrayList();//["thin", "thick"]
boolean hasNN = false, hasJJ = false;//node nn ve jj varmi

        // Traverse the Children
        for (int i = 0; i < trChildren.length; i++) {
              Tree currentTree = trChildren[i];//each node also is a
tree
              strLabels[i] = traverseTree(currentTree);//recursive
function call go back to begnning.
              String sp = strLabels[i].split(":")[0];

              if( sp.compareTo("NN") == 0 || sp.compareTo("NNS") == 0
|| sp.compareTo("NNP") == 0  ){
                    listNN.add(
strLabels[i].split(":")[1].toLowerCase() );
                    hasNN = true;
              }else if( sp.compareTo("JJ") == 0 || sp.compareTo("JJR")
== 0 || sp.compareTo("JJS") == 0){
                    listJJ.add(
strLabels[i].split(":")[1].toLowerCase() );
                    hasJJ = true;
              }
        }

        //Tag and Adjective exits at the same time
        if( hasNN && hasJJ ){

          if( this.isToken( listNN , listJJ ) == true ){
          this.hasRequiredTags = true;//At least one tags in sentence

                System.out.print( labelNode + ": " );
                for (int i = 0; i < trChildren.length; i++) {
                  System.out.print( strLabels[i] + " ");
                }
                System.out.println();
            }
        }
if( labelNode.compareTo("NN") == 0  || labelNode.compareTo("NNS") == 0
|| labelNode.compareTo("NNP") == 0 ||labelNode.compareTo("JJ") == 0
||  labelNode.compareTo("JJR") == 0 || labelNode.compareTo("JJS") == 0
){
            try{
                  labelNode = labelNode + ":" +
node.firstChild().label();
              }
              catch(NullPointerException exc){

              }
        }

        return labelNode + ":" + node.children().length;
    }
     // isToken (nn , jj)

private boolean  isToken(ArrayList tokenArray, ArrayList adjArray){
        int j = 0, i = 0;

        for(i=0, j=0; i<tokenArray.size() ; i++){
        }
```

```
for( int TagIteretor = 0; TagIteretor < myTags.length; TagIteretor++ ){
            String[] phrase = myTags[ TagIteretor ].split(" ");

for(i=0, j=0; i<tokenArray.size() && j< phrase.length; i++){
if(tokenArray.get(i).toString().indexOf(phrase[j].toLowerCase()) == 0 )
                {
                    if (i == phrase.length-1)
                    {

       this.tempValues[TagIteretor] = adjArray.get(0).toString();


                        if( IsStringExistInArray(listArray[TagIteretor],
this.tempValues[TagIteretor]) == false){
                            listArray[TagIteretor].add(
this.tempValues[TagIteretor] );
                            return true;
                    }
                    j++;
                }
                else
                {
                    //System.out.print(",? ");
                    if (j != 0) {
                        i--;
                    }
                    j = 0;
                }
            }

        }
        return false;
   }

    public void AppendTempValuesToDataResult()

{//dataResult<StringBuilder> appends
if( this.hasRequiredTags == false ) //Sentence does not have the tags

{             return;
        }

        for(int i=0; i<this.tempValues.length; i++){
            //if activetags =false(The tag does not have adjective)
            if( activeTags[i]  != true ){
               continue;//break if  increment index and go back to loop
            }

            System.out.print( this.tempValues[i]  );
            dataResult.append( this.tempValues[i] );
            // Remove last semicolon
            if( this.tempValues.length - 1 != i ){

                System.out.print(", ");
                dataResult.append(", ");
            }
        }

       System.out.println();
       dataResult.append("\n");
```

```
    }


   public static void RemoveEmptyTags(){

        for( int i=0; i<listArray.length; i++){

            if( listArray[i].size() == 0  ){
                activeTags[i] = false;
            }
        }
    };

//This function checks if the same adjective exists in adjective list
//It is written to prevent redundancy.
public boolean IsStringExistInArray(ArrayList adjList,String adjValue){
   // Find Duplicate adjective if any
  for(int index = 0; index < adjList.size(); index++){
       if(adjList.get( index ).toString().compareTo( adjValue )  == 0){
                return true;// Duplicate
            }
        }
        return false;
   }

   public static String RemoveWhiteSpace(String tagName){
   //Remove whitespace from 2 word tag  : heat transfer -> heattransfer
WEKA requirement
        String[] subTags = tagName.split(" ");
        tagName = "";
        for(int index = 0; index < subTags.length ;index++){
            tagName += subTags[index];
        }
        return tagName;
   }

//This function creates the WekaResult.arff text file in the working
directory and formats the for WEKA
   public static void PrintResult(){
        Writer output = null;
        try {
            File aFile = new File("WekaResult.arff");
            output = new BufferedWriter(new FileWriter(aFile));

            System.out.println("@relation heattransfer.symbolic");
            System.out.println();
            output.write( "@relation heattransfer.symbolic\n\n" );

            for (int TagIteretor = 0; TagIteretor < myTags.length;
TagIteretor++) {
                if( activeTags[TagIteretor]  != true ){
                    continue;
                }
                ArrayList list = listArray[TagIteretor];
                System.out.print("@attribute " +
RemoveWhiteSpace(myTags[TagIteretor]) + " {");
                output.write( "@attribute " +
RemoveWhiteSpace(myTags[TagIteretor]) + " {" );
                for (int i = 0; i < list.size(); i++) {
```

```
                    System.out.print(list.get(i).toString());
                    output.write( list.get(i).toString() );
                    //Removing the Last semicolon from .arff file
                    if (list.size() - 1 != i) {
                        System.out.print(", ");
                        output.write( ", " );
                    }
                }
                System.out.println(" }");
                output.write(" }\n" );
            }
            System.out.println();
            System.out.println("@data");
            System.out.println(dataResult.toString());
            output.write( "\n@data\n" );
            output.write( dataResult.toString() );
        } catch (IOException ex) {

Logger.getLogger(XmlBuilder.class.getName()).log(Level.SEVERE, null,
ex);
        } finally {
            try {
                output.close();
            } catch (IOException ex) {

Logger.getLogger(XmlBuilder.class.getName()).log(Level.SEVERE, null,
ex);
            }
        }
  }
```

APPENDIX B:

SAMPLE WEKA FILE

@relation heattransfer.symbolic

@attribute Agitation {high, fluid, enhancing, higher}
@attribute Alloy {metallic, various, liquid, ferrous, specific}
@attribute CarbonContent {base}
@attribute CoolingRate {typical, critical}
@attribute cracking {quench}
@attribute Distortion {mechanical, greater, more, less, warping}
@attribute Fixture {different, specimen, quench}
@attribute Geometry {pan, different}
@attribute Hardness {tempered}
@attribute HeatTransferCoefficient {average, high, calculated, higher, sur, effective, final, meaningful}
@attribute Oxide {thin}
@attribute Part {quenched, carburized, various, heat-treated, integral, finished, heated, single, intensively}
@attribute Quenchant {liquid, same, concerned, different, specific}
@attribute Quench {different, real, commercial, liscic-nanmac, agitated, portable, ideal, still-bath, water-base, fast, slowest, dint-rent, standard, lite, austempering, spent, nitrate-nitrite, rapid, intensive, high-velocity, atotal, continued }
@attribute QuenchSeverity {earlier, new, molten}
@attribute Size {standard}
@attribute Surface {very, flat, top, liscic-nanmac, certain, hot, hard, curved, rapid, different, other, bottom}
@attribute Temperature {various, different, fast, 304, higher, lower, fluid, specified, high, quench, quenchant }
@attribute Type {special, certain, various}

@data
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, thin, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, average, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, carburized, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, different, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, various, ?,
?, ?, ?, ?, quench, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, various, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, different, ?,
?, ?, ?, ?, ?, ?, different, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, real, ?, ?, ?, ?, ?,
high, ?, ?, ?, ?, ?, ?, ?, ?, high, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, earlier, ?, ?, ?, ?,
?, ?, ?, ?, ?, mechanical, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, different, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, calculated, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, new, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, quenched, ?, ?, ?, ?, ?, ?, ?,

?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, commercial, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, quenched, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, different, ?, ?, ?, ?, ?,
?, metallic, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, liquid, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, liquid, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, special,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, very, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, fast, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, flat, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, same, different, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, 304, ?,
fluid, ?,
fluid, ?,
fluid, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, top, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, agitated, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, liscic-nanmac, ?, ?,
enhancing, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, concerned, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, molten, ?, ?, ?, certain,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, new, ?, ?, ?, ?,
?, ?, ?, typical, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, different, portable, ?, ?, ?, ?, ?,
?, various, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, critical, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, lower, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ideal, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, still-bath, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, water-base, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, certain, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, different, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, higher, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, different, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, fast, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, slowest, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, dint-rent, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, hot, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, hot, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, standard, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, pan, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, quenched, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, fluid, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, sur, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, lite, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, higher, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, effective, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, different, ?, ?, ?, ?, ?,
?, liquid, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,

higher, ?, ?, ?, ?, more, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, different, ?, ?, ?, ?, ?, different, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, various,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, specific, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, heat-treated, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, tempered, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, austempering, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, specified, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, spent, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, high, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, integral, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, hard, ?, ?,
?, ?, ?, ?, ?, less, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, nitrate-nitrite, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, finished, ?, ?, ?, ?, ?, ?, ?,
?, ferrous, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, specific, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, quench, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, quench, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, quench, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, rapid, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, standard, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, flat, ?, ?,
?, ?, ?, ?, ?, ?, specimen, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, specimen, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, curved, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, heated, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, single, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, intensive, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, intensive, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, intensive, ?, ?, rapid, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, intensive, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, intensive, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, intensively, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, intensive, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, quenchant, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, intensive, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, top, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, different, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, intensive, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, high-velocity, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, atotal, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, final, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, intensive, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, intensive, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, other, ?, ?,
?, ?, base, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, continued, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, intensive, ?, ?, ?, ?, ?,

?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, intensive, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, intensive, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, meaningful, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, intensive, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, top, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, different, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, intensive, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, warping, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, bottom, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, bottom, ?, ?,
?, ?, ?, ?, ?, ?, quench, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,

REFERENCES

[1] A. Fernández-Valmayor, F. M. Baltasar, A. Navarro, J. L. Sierra. "Building Applications with Domain-Specific Markup Languages: A Systematic Approach to the Development of XML-Based Software, SpringerLink, January 01, 2003.

[2] A. S. Varde, E. A. Rundensteiner, M. Maniruzzaman and R. D. Sisson Jr. Quenchml: The quenching markup language, 2002.

[3] A. S. Varde, E. A. Rundensteiner, M. Maniruzzaman and R. D. Sisson Jr., Estimating Heat Transfer Coefficients as a Function of Temperature by Data Mining, Technical Report , Worcester Polytechnic Institute (WPI),Worcester, MA.

[4] A. S. Varde, M. Maniruzzaman, E. A. Rundensteiner and R. D. Sisson Jr., The QuenchMiner™ Expert System for Quenching and Distortion Control , Worcester Polytechnic Institute (WPI),Worcester, MA.

[5] A.S. Varde, M. Maniruzzaman, E. A. Rundensteiner and R. D. Sisson Jr., Estimating Heat Transfer Coefficients as a Function of Temperature by Data Mining, Technical Report, Worcester Polytechnic Institute (WPI), Worcester, MA.

[6] AHanus, Heat Treatment Plant for Aluminum and Quenching Process, pdf.

[7] A. Taylor, The Penn Treebank Project, http://www.cis.upenn.edu/~treebank/.

[8] B. McCartney, The Stanford Parser: A statistical parser, http://nlp.stanford.edu/software/lex-parser.shtml, 2003.

[9] Cognitive Science Laboratory at Princeton University, Wordnet: A Lexical Database for the English Language, http://www.clrn.org/weblinks/details.cfm?id=2238.

[10] E. Rilloff, An empirical study of automated dictionary construction for information extraction in three domains, Artificial Intelligence Journal, 1996.

[11] IBM, Intelligent miner for text, http://www3.ibm.com/software/data/iminer/fortext/, 2000.

[12] I. Fonteechlc, M. Maniruzzaman and R.D. Sisson, Jr., Optimization of an Aluminum Alloy Quenching Process in Polyalkylene Glycol Polymer Solution Using Taguchi Method, Technical Report, CHTE, Materials Science and Engineering Program Mechanical Engineering Department ,Worcester Polytechnic Institute (WPI),Worcester, MA, October 2002.

[13] J. Kang, L .Zhang, R. Purushothaman and Y. Rong, Computer-aided Heat Treatment Planning System for Quenching and Tempering (CHT–q/t),

http://www.wpi.edu/Pubs/ETD/Available/etd-0830102-113605/unrestricted/vader.pdf, June 2002

[14] J. Thomere, K. Barker, V. Chaudhri, P. Clark, M. Eriksen, S. Mishra, B. Porter, and A. Rodriguez, A web-based ontology browsing and editing system, In AAAI, 2002.

[15] L. Singh, P. Scheuermann and B. Chen, Generating association rules from semi-structured documents using an extended concept hierarchy, Conference Knowledge Management, pages 193–200, 1997.

[16] M. Fontecchio, M. Maniruzzaman and R. D. Sisson, Jr., Quench Factor Analysis and Heat Transfer Coefficient Calculations for 6061 Aluminum Alloy Probes Quenched in Distilled Water, Technical Report, Materials Science and Engineering Program Mechanical Engineering Department, Worcester Polytechnic Institute, Worcester, MA

[17] M. Maniruzzaman, R. D. Sisson, Jr., Gas Quenching: An Environment Friendly Heat Treatment Process, Technical Report, WPI, Worcester, MA, 2006

[18] M. Maniruzzaman, R. D. Sisson, Jr., S. Ma, Characterization of the performance of mineral oil based quenchants using the CHTE Quench Probe System, Technical Report, Materials Science and Engineering Program Mechanical Engineering Department (WPI),Worcester, MA, 2002

[19] M. Maniruzzaman, R. D. Sisson, Jr., Optimization Gas Quenching - An Environment Friendly Heat Treatment Process, Technical Report, CHTE, MPI-WPI, Worcester, MA, October 2002

[20] M. Reed, A Definition of Data Warehousing, http://www.intranetjournal.com/features/datawarehousing.html, 2002

[21] P. J. Imielinski, Introduction to Expert Systems, Addison Wesley Longman, Reading, Massachusetts, 1999.

[22] R. Agrawal, T. Imielinski and A. Swami, Mining Association Rules between Sets of Items in Large Databases, *SIGMOD*, **22**(2):207-16, June 1993.

[23] R. D. Sisson Jr., M. Maniruzzaman, A. S. Varde, D. K. Rondeau, M. Takahashi and S. Ma, Quenching-Understanding, Controlling and Optimizing the Process, Technical Report, Worcester Polytechnic Institute (WPI),Worcester, MA, May, 2003

[24] R. Kirkby, Attribute-Relation File Format (ARFF). http://www.cs.waikato.ac.nz/~ml/weka/arff.html, April 1st, 2002

[25] S. Chaudhari and U. Dayal, An overview of Data Warehousing and Olap Technology, SIGMOD Record, 26(1):65–74, 1997.

[26] S. Ma, D. K. Rondeau, M. Takahashi, M. Maniruzzaman, R. D. Sisson Jr. and A.S. Varde, Quenching-Understanding, Controlling and Optimizing the Process, Technical Report, Worcester Polytechnic Institute (WPI),Worcester, MA, May, 2003

[27] Stephen Soderland. Learning to extract text-based information from the World Wide Web. In Third International Conference on Knowledge Discovery and Data Mining, 1997.