CRITICAL INFORMATION RETRIEVAL FROM EMAILS

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Souvik Sen

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

November 2014

Fargo, North Dakota

North Dakota State University
Graduate School

Title

CRITICAL INFORMATION RETREIVAL FROM EMAILS

By

Souvik Sen

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Juan (Jen) Li
Chair

Dr. Jun Kong

Dr. Gursimran Walia

Dr. Ying Huang

Approved:

11/19/2014	Dr. Brian M. Slator
Date	Department Chair

ABSTRACT

With efficiency being a driving force in today's ecommerce, emails have become a major form of communication. However, deciphering information from these emails has been a be-labored task. With every email containing proprietary information, handling this information has become an arduous task that requires money, time and effort. To tackle this ecommerce problem, a computerized solution is important to expedite the extraction of information. In this paper, we have applied Named Entity Recognition, different rules and algorithms to extract important information from emails. The proposed solution tackles challenges revolving around tabular and natural language formats, which are the largest formats used for email communication. Use of this solution makes business easier to navigate through a variety of attachments: PDF, Word, and Excel. The proposed application has been applied on a dataset supplied by a Transportation company by which the results have been captured.

ACKNOWLEDGEMENTS

I wish to express my deep sense of respect, and indebtedness to my adviser, Dr. Juan (Jen) Li, for her valuable suggestions, and ceaseless encouragement during the project work. I would also like to thank the committee member. I have had a great pleasure in expressing my sincere thanks to my advisory committee members Dr. Jun Kong, Dr. Gursimran Walia, Dr. Ying Huang, for their valuable support. Last but not least, I would like to thank my fiancé Olimpiya, my caring parents, my other family members and my friends for their inspiration, encouragements, everlasting blessing, and abundant love for me for the successful completion of this achievement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

The concept of email came about 50 years ago; it was MIT's "Compatible Time-Sharing System, which has been built for multi users, who had the ability to login into a centralized system to store and share important documents for remote access [12]. Since then the popularity of email has been increased in a rapid manner. Over the past five decades, Email has become one of the easiest and reliable modes of communication, mainly because of its efficiency, low cost and support for wide range of information [1]. Recent studies show that email is still number one online activity though there are new concepts like social networking [3]. When it comes for the corporate users, radicate surveys say that corporate users send and receive about 110 messages [5] per day in average and out of them one third are messages sent. Those statistics are quite constant and has not been changed too much in last decade [6]. A report from 2003 [7] shows that about 80% of the business users prefer email communication over others for their work purpose. A report from 2008 [8] says that 62% of the employees in United States can be considered as Networked Workers as they use internet and email on their work on daily basis. Information generated by business entities can be considered as highly useful asset based on how well it is managed. Email is not different here [9]. Email is now essential for many of the common industrial [9, 10, 11] functions like task management, collaboration, generating alerts, archiving and interoperability.

It is pretty common for many of the organizations to receive product or service requests via email. These emails are required to be read and collect significant information to process that request. The processing of those requests will be correct, it essential information will be collected in a correct manner. Companies receive thousands of such important emails every day. Hence to process them in a short time with maximum efficiency, an automated system is really essential. This program will extract and save the featured information in a database which will be consulted to provide the necessary service. For example: a computer servicing company receives piles of emails form the customers related to their computer hardware or software issues. To serve the clients better and faster, the company needs to know the details about the requested services on time and with correct information. The details about the service the customer is looking for can be like what the service is related to, where the service is required, when the service required etc. Email information extraction program will extract all of this information as correctly as possible and will save them in the company database for further service. Company then uses the database entries to figure out the necessary solution for that. Time is another issue here; some services are really important and required urgently. Manual reading emails and providing service can take a long time and user can suffer because of that. Hence an automatic email extraction procedure will surely solve that time delay issue.

Many ecommerce companies record email receipts of online transactions which are full of essential product information including product category, price, date of purchase etc. If this information can be extracted and saved in a good manner, it can be used for several purposes including a recommendation system [2]. If the system can identify the type of product a specific user is buying, then the system can suggest further products to that user using extracted information.

1.1. Email Sender Analysis

In email analysis, it is very important to analyze the sender of the email first. It is pretty common in internet world to get spam emails. Lots of people and organizations try to fraud others via emails. Hence highly secured industries need a trusted email data handling process; otherwise it will be easy to cheat them with false service request. Sometime it is essential to band certain emails and email senders for specific purpose and hence email sender information analysis is of high importance here. As an example we can say, suppose some customers (person or organization) are known for their false claims about the company product service and quality. Full processing all those false emails is waste of resource for the organization. So it is important for the company to black list those people or organizations for farther misconduct and wrong complain about the company. In another case sender analysis will help to serve the customer faster. Often a particular organization sends email only for a specific kind of service. Company can make a list of well-known customers and that will help them to identify them at front by sender email addresses. Email service processing will become easy for them. In our project we have used Dictionary matching on email addresses to get a match with the Blacklisted or well-known email addresses and domains.

1.2. Email Categorization

Categorization of emails becomes really essential when the number of emails someone is getting is huge [1]. Proper email categorization can help a company to do their work much smoother and organized way. Lack of proper email categorization can create massive negative effect on personal and company performance; proved by Dabbish and Kraut in 2006 [1]. Hence automatic email categorization can be really productive for farther processing of emails. Based on the email categories, different processing can be done on the emails in the system. For example: in a Truck allocation company, it will be really helpful if they can know whether a specific request is for asking truck availability or asking a load. If that can be figured out, then serving that request will be much easier.

1.3. Email Content Types

Email content can be of various types: normal textual data, tabular data and no specific format. Textual data are mainly Natural Languages i.e. formal English sentences and I am mainly focusing on this in my project. Some emails contain attachments like pdf, word files, excel files etc. These pdfs, word files or excel files can contain important information for the organizations, so processing them are required. In my project I have created a system which extracts Pdf and excel attachments from the email, pre-processes them and then extracts information from them. To make the email extraction process simple, I have converted all these files in one text format and then used Named Entity Recognition and Decision-Tree Algorithms.

1.4. Content Analysis and Extraction from the Emails

There are different challenges in extraction of exact texts in email content. Sometimes emails are neither written following proper grammatical rules not spellings. So to get the information correctly we have to handle them properly defining different rules in our project. Apart from that emails contain emails dates, locations, times, some numeric values etc. in different and multiple formats. So processing all of them and extract them correctly is a challenge and we have countered them in our project. For example: 08/20/2014 and 20 Aug, 2014 are same dates however they have appeared in different formats, so they are required to be extracted and filled up in the database with a single format.

Here in this project, I have used a set of emails from a company named ValleyExpress as our dataset. ValleyExpress is a nationwide freight company specializing in temperature control shipments, managing a fleet of more than 130 units. They receive tons of emails, requesting allocation of trucks for available loads or vice-versa. I have taken a set of emails with Natural Language body content and PDF or excel as attachment to test my system.

CHAPTER 2. BACKGROUND AND RELATED WORK

2.1. Natural Language Processing: Named Entity Recognition

Natural Language Processing is a machine learning or specifically statistical machine learning technique [14]. With the help of NLP we can analyze, understand, and generate languages that humans use naturally. Hence we will be able to address any computer as though we are addressing another person with that generated language.

In Information Extraction (IE), there are several advanced systems available, like GATE [17], KIM [18], C-PANKOW [19] etc. which can perform data extraction quite efficiently. However C-PANKOW [36] or knowItAll [37] works well on general information domain like web but this is not applicable to the enterprise specific content. So to get the best result in information data extraction, Natural Language is really essential. NLP techniques decompose the sentences and tag the parts of speech like noun, verb, adjective, and pronoun. Then different information retrieval techniques can be applied on the result. A point to remember is that NLP techniques are easily available in language like English but not in other languages. Areas where domain specific knowledge is important, Natural Language Processing is difficult to use [20]. There are many well-known parts of NLP in Computer Science like Morphological segmentation, Word sense disambiguation, but for this project our interest is in Named entity recognition.

2.1.1. Named Entity Recognition

"Named entity recognition is a subtask of Information extraction" [15]. Named entities are phrases which contains a person's name, company name, location etc. For example: *"Mr. Andrew works for Medtronic since July 21, 2010"*.

The above sentence contains 3 named entities "*Mr. Andrew"* a Name, "*Medtronic"* and organization and "*July 21, 2010"* a time.

This Named Entity Recognition or NER is the process of information extraction which tries to find elements in the text into pre-defined categories like person name, company name, location, date, time, numerical value etc. [15]. NER performs surface parsing, delimiting sequences of tokens that answer questions like "who", "where", "how much" in a sentence to determine answers for person or organization name, locations and quantities from that sentence. NER can be used as the first step in a chain of processes. Next level of processing could relate two or more NEs or even give semantics to that relationship using a verb [16]. Named Entity Recognition is a kind of Statistical Machine learning modeling and it requires a large amount of annotated training dataset. Hence a dataset needs to be annotated using some Annotator and that has to be feed into NER model as a training dataset. There are lots of pre-annotated dataset already available in the web and those can be used as training dataset as well. The performance and accuracy of the techniques depends upon the quality of the training data that has been provided to the parser.

In NER and NLP is an open research area and multiple research is still running. In past and even now most of the statistical analysis in Natural Language Processing has been performed using local structure for a dataset. For example Hidden Markov Models (Leek, 1997; Freitag and McCallum, 1999), Conditional Markov Model (Borthwick, 1999), and Conditional Random Fields (CRFs) (Lafferty et al. 2001) are recent applications in Information Extraction and all of these are based on Markov property; which says that the state at a particular position in a sequence depends on a local window. However in a research in Stanford University NLP group, they have used non-local processing [21] to get better result in most of the cases as Natural

Language contains a great deal of non-local structure. Here they have used a simple Monte Carlo algorithm "Gibbs sampling". This is perfect for inference in any factored probabilistic model, including sequence models. They have provided an example given in Figure#1and demonstrated benefit of non-local processing in NLP through that. In example showed in Figure 1, the 2nd occurrence of the token "Tanjug" is mislabeled by CRF based NER techniques. The 1st occurrence of the field shows that "Tanjug" is an Organization however based on the local processing of data; the system gets confused whether "Tanjug" is an Organization or a person's name. In this paper the author proved that this can be determined using non-local processing of NLP.

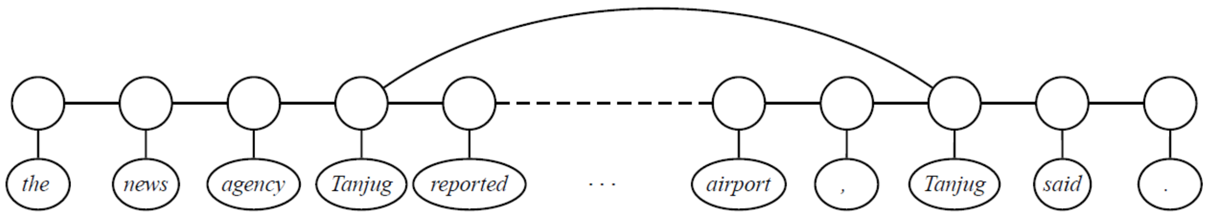


Figure 1: Example of a Label Consistency Problem from CoNLL 2003 English Dataset

For our project as we want to find specific information from emails and Named Entity Recognition is best suited here. We have used Stanford's Named Entity Recognition (NER) to extract important information like Person Name, Organization Name, and Date-Time etc. as this gives good result for this domain and easily usable.

2.2. Key Value Pair Based Information Extraction

This is one of the most common used techniques in Information Extraction. The main concept key-value pair is, there will be a key (an object type) and an associated value (Matched text) for that key [9]. A gazetteer is a geographical dictionary or directory used in conjunction with a map or atlas [23]. This gazetteer is used to extract key-value pairs from a text. For example:

"Mr. Andrew has a shipment to send from Minneapolis, MN to Omaha, NE on 5th July, 2014"

Here key: person name and the corresponding value is Mr. Andrew,

Key: Location Value: Minneapolis, Omaha,

Key: Date Value: 5th July, 2014

We can extract all possible key-value pairs from a text using defined words and gazetteers and these key-value pairs can be used for farther processing. In our project we have set many rules based on the key-value pairs and keywords to get more complicated information from a text. Based on some of these rules we can detect which location is source and which one is destination.

2.3. Dictionary Matching

Dictionary matching is a commonly used operation in Information Extraction [24]. A predefined dictionary is used to find different important keywords in a set of strings. For example Dictionary matching can be used to find Months of a calendar year, States etc. String searching is used to match with Dictionary entries. Aho-Corasick string matching algorithm is an algorithm which helps finding elements of a Dictionary of strings within an input text. Let us posit that we have to locate any pattern from a set $P = \{P1, \ldots, P_k\}$, in a target set $T[1\ldots m]$. Now let us consider $n = \sum_{i=1}^{n} |Pi|$. It works in time $O(n + m + z)$, where z is number of pattern occurrences in T [25]. In this Algorithm, the system generates a finite state machine and that is like a keyword tree or known as a tree. This tree is constructed with patterns P.

1. Each edge of the tree is a character

2. Every two edges out of a node have separate labels

   Label of a node v = concatenation of edge labels = L(v)

3. for every pattern in P, there exists a node v as L(v) = P

4. L(v) of any leaf v = p belongs to pattern set

   For example a keyword tree {he, she, his, hers} [25]

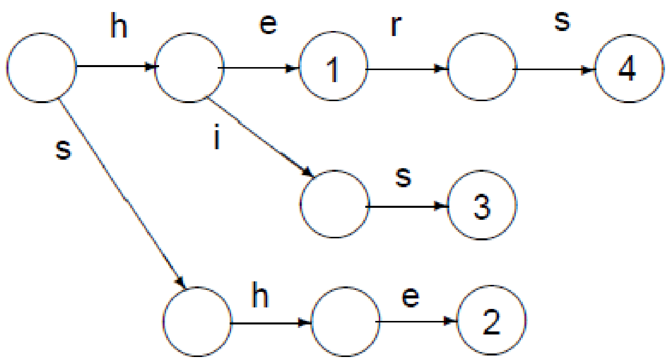


Figure 2: Pattern Matching Example

Keyword lookup: The lookup starts from the root of the tree. Then follow the route labeled by characters of P as long as possible. In Figure 2 we have presented an example of pattern matching tree.

- If a route is found that leads to an identifier then that pattern belongs to that set of identifier.
- If the route finishes before the pattern completes, then that pattern does not belong to that Dictionary.

2.4. Related Work

As we have discussed earlier that emails are very common medium of electronic communication for almost last 40 years, a considerable amount of research has been done on this field to get benefit from those email data. In a paper about extracting product information from

Emails Receipts, author has presented an efficient algorithm based on Markov Logic [26]. Markov logic is combination or probability and logic. In this work they have encountered many challenges like E-receipts can be generated from different templates. So making a generalized rule is always challenging. Maximum of the E-receipts are based on plain text instead of HTML tagging and that makes the process of information extraction much more complex as data representation is irregular. They have created a corpus of unlabeled E-receipts and they have identified all possible templates by jointly clustering all those E-receipts [27].

In another study in University of Ottawa, Canada, [28] people have used semantic tagging and domain knowledge for the enterprise to extract information form an outgoing email in a company. These extracted data they are using for the purpose of detecting the privacy risk of the organization by matching them against a set of Compliance rules.

In 2010 in Institute of Informatics Slovak Academy of Science, people presented a paper [9], which highlighted Email Analysis for Enterprise Benefit. They have proposed a light-weight process using different NLP techniques like Named Entity Recognition (NER), Coreference Resolution (CO), Template Element Construction (TE), Template Relation Construction (TR) and Scenario Template Production (ST), then Key-Value pair based information extraction to get the important information regarding enterprise emails. Now those extracted information has been processed using Semantic Trees, Email Social Networks, and Graph Inference to get the job done. On the other hand Email protocols and Email server have been used in this process.

CHAPTER 3. SYSTEM DESIGN

3.1. Domain Knowledge and Ontology

Prior to the inception of email information extraction and subsequent processing, it is essential to acquire a concrete domain knowledge which basically captures specific information about the structure of the organization, nature of the users who will be using the system and their corresponding expectations. An intensive analysis of the company database elicits deep domain knowledge. This domain knowledge can be used in training our Named Entity Recognizer models and also in creating rules to extract information. For example a University consists of students, professors, office staff and many other members. In the University the students can be characterized by some well-defined attributes some of which can be Student_Id, Student_Name, and Department. Hence the domain knowledge is really important in order for the system to handle emails related to a University more efficiently and predict accurate results. We can acquire Domain knowledge from two sources- first from Organization Databases and Dictionaries and next from Domain ontology. The latter is required to represent items in a tree or a hierarchical structure [28]. In Table 1, we have presented some Tables for email categories and attributes of the Tables.

Table 1: Example Tables for Truck and Load Delivery Company

Objects	Attributes
Blacklisted emails	Senders Name, email id, email domain
Load emails	Load Type, Source Location, Destination Location, Date
Truck emails	Truck Type, Available location, Available Date

The Domain ontology is a hierarchical organization of roles from a database [28]. There is a clear Mapping between entries in the database and the roles represented in a hierarchical Ontology. In Figure 3 we have given a hierarchical ontology of email categorization based on the Database tables shown in Table 1.

3.2. Email Preprocessing

An email contains multiple parts in Figure 4 has displayed all those possible models with an example email. We are processing each module at a time to extract important information from each of them. The process flow for the entire system has been depicted in Figure 5.

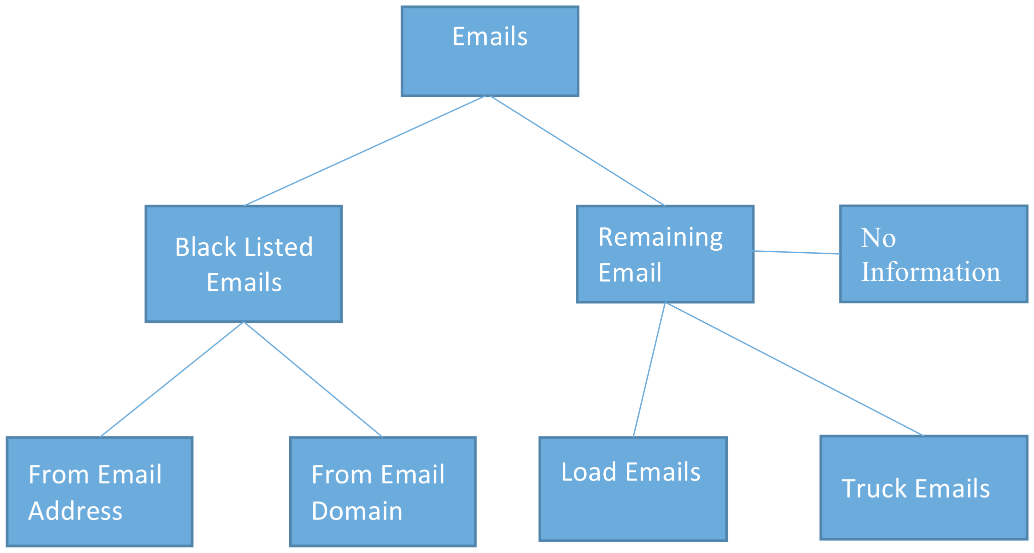
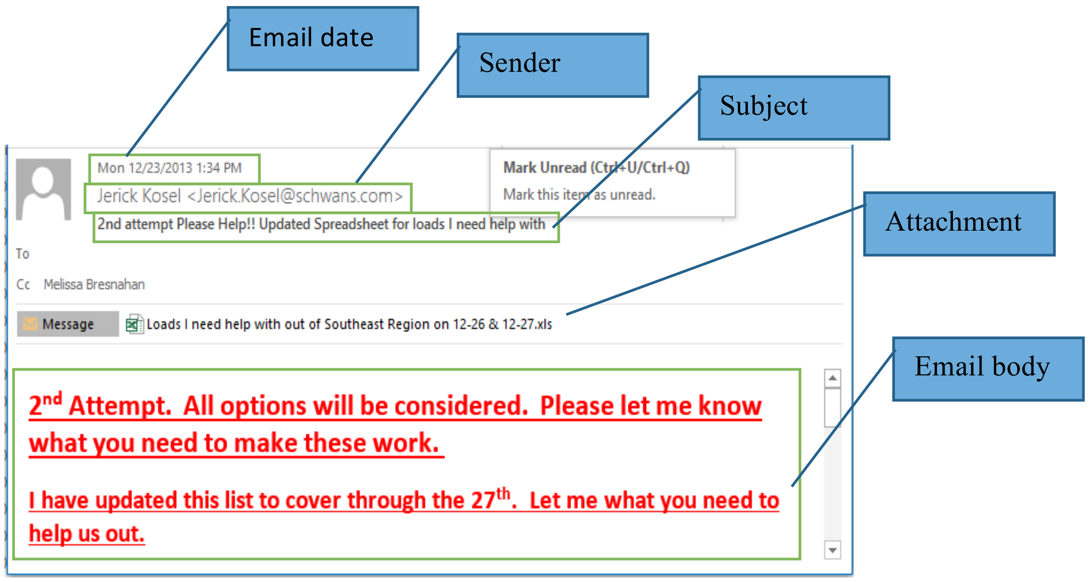


Figure 3: Example of Hierarchical Ontology of Email Categorization

3.2.1. Sender Extraction and Analysis

The email client receives emails from different email addresses which are being preserved by the email client for future references. Now the first part of our process is Email sender extraction. In our work we have considered that the email client already does this extraction and the sender email address is available to us.



Mon 12/23/2013 1:34 PM

Jerick Kosel <Jerick.Kosel@schwans.com>

2nd attempt Please Help!! Updated Spreadsheet for loads I need help with

Cc Melissa Bresnahan

Loads I need help with out of Southeast Region on 12-26 & 12-27.xls

2nd Attempt. All options will be considered. Please let me know what you need to make these work.

I have updated this list to cover through the 27th. Let me what you need to help us out.

Figure 4: Example Email Fields

We perform some specific processing steps on the email addresses gathered from the email client. Based on the organizations, we can have a collection of blacklisted email senders. Blacklisted email senders refer to people or organizations who have been identified as unwanted by the organization under consideration. We posit that the company already possesses this list and it is accessible to us. If we find any match with the blacklisted email addresses while processing the received email addresses, we ignore those emails without further processing.

In course of processing email addresses, we split them by "@" and then extract the email domain to determine blacklisted email domains. We use simple string matching techniques to find out the match with the Dictionary of blacklisted email domains or email addresses. For example, we have a blacklisted email domain as "factory.com". The company might have received an email from james_dube@Factory.com. When we process this email and split it by the character "@", we get that the Domain name is Factory.com. In this case a positive hit is encountered during the process of extracted keyword comparison with the blacklisted email domains as a consequence of which we skip the particular email.

This preprocessing of the emails before processing their email body or attachments saves a lot of CPU processing time and resources. This also contributes towards enhancing the security features of organizations by safeguarding their internal operations from the attacks of fraudulent users with malicious intent.

3.2.2. Subject Extraction, Analysis and Categorization

Email subject plays a significant role in knowing the purpose of the emails. In this work we obtain the email subjects from the email clients in the form of strings. We use different rules on the subject lines of the emails to know their purpose. We use those rules to construct a decision-tree based categorization algorithm to categorize emails. We apply domain knowledge of the organization to create specific rules attributed towards identifying the differences between emails. For instance we conducted an elaborate study with the emails related to a Freight Delivery company. Based on the email subjects, we try to detect whether the email has been received from a client who wants a truck or from a company which has trucks available to transfer loads. This email categorization is really important before further processing. Based on the type of organization we are working with, it can be a required information that has to be provided. Apart from that if we can categorize the emails first, then we can use category specific algorithms for further processing. This approach has the advantage of saving considerable CPU processing time thus making the email extraction process to be executed in a much faster time.

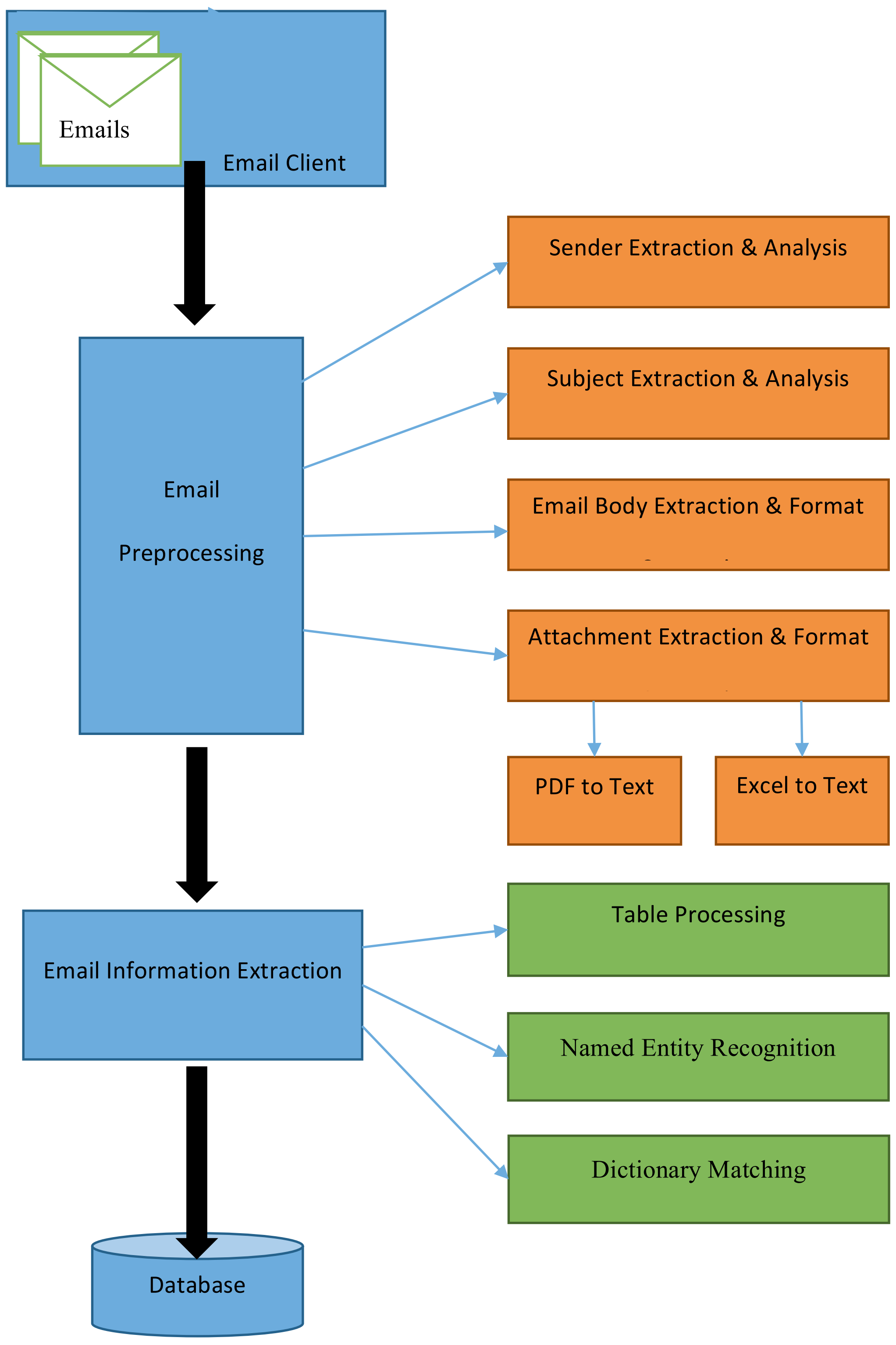

Emails
Email Client
Email Preprocessing
Sender Extraction & Analysis
Subject Extraction & Analysis
Email Body Extraction & Format
Attachment Extraction & Format
PDF to Text
Excel to Text
Email Information Extraction
Table Processing
Named Entity Recognition
Dictionary Matching
Database

Figure 5: System Diagram for Flow of the Processes in the System

A challenge in this approach emerges from the fact that some emails do not include email subjects or even if they do, the subjects fail to succinctly represent the information in the email that can be used to separate it from the others. In such cases analyzing the entire email text becomes essential to categorize them. In this context we have created a set of rules based on some training sample emails and domain knowledge. We use these rules first on the email subject content and attempt to identify the purpose of the email and if we can categorize it based upon the result. In cases where the application of these rules on the subject content fails to achieve classification, we apply similar rules on the email body content or attachment contents. Based on the study of the sample emails in our used dataset, in more than 90% of the cases we were able to find the purpose of the email and could achieve email categorization successfully.

Rules for decision-tree based categorization can be constructed from domain specific keywords. We accumulated a set of emails from the company which has built our training dataset. A careful study of these emails in the next step generated specific categories for them. This study revealed a list of keywords in combination with their specific orders for categorization. From the integration of the keywords and their orders we were able to generate the set of rules. In the subsequent step we map these rules to their specific email categories. This implies that if rule a matches to a particular email, then it can be concluded that it belongs to category A.

Based on the above discussions, we can describe algorithm of decision-tree based categorization of the emails in the form of the following sequential steps of processes:

Algorithm: Email Categorization

1. START
2. Gather **Domain knowledge** and study sample emails for the organization

```
3.  Generate list of keywords
4.  Create rules based on those keywords and their order of appearance in the text
5.  Create set of categories for the emails
6.  Create a Mapping between rules and categories
7.  Study email subjects and apply these rules
8.  If Category received then
9.      Done. Go for next level of processing
10. Else
11.     Apply rules on the email body content or email attachments
12.         If Category received then
13.             Done. Go for next level of processing
14.         Else
15.             Categorization failed.
16.         End if
17. End if
18. END
```

3.2.3. Attachments Extraction and Format Conversion

Email client extracts the attachments, saves it in a memory location and sends the link to the system. It is considered that we have access to all these file paths. We can separate these email attachments based on their file extension. In our work we have encountered PDF files and excel files as attachments with the emails in this system. If received emails have attachments of types Excel or PDF we are able to process them. Attachment processing initiates from identifying the type of the available attachment. According to their types successive processing is performed. In the Figure 6 we have depicted the flow of the process.

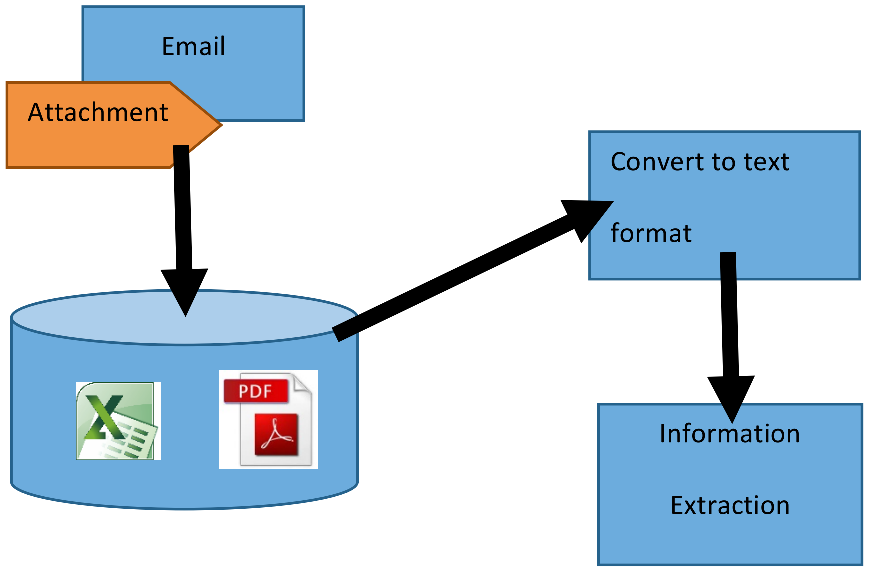


Figure 6: Attachment Processing Flow

3.2.3.1. PDF to Text Conversion

We have encountered two kinds of PDF files in our project. One kind of PDF files contains Natural Language content and another kind contains tabular data in it. This makes it difficult to extract and convert the format of PDF files to text forms by following a single process. Hence we have adopted two different algorithms to process these two formats.

We apply table processing algorithm first in order to determine which algorithm should be applied on a PDF file. If the content of the PDF is Table then it returns the Text format of the PDF. If the content is not table then it notifies the system about it after which the next algorithm is applied for processing.

Plain Text: For Plain Text PDF files, we have used Apache's **PDFBox** [29] library methods to convert the PDF files to text files. We have used the PDModel concept here.

PDModel helps creating and manipulating PDF Documents [35]. PDDocument class in PDModel represents the in-memory representation of a PDF file. By using this PDDocument class we were able to access the attributes of a PDF Document.

We have extracted the text from the PDF by regions. Using PDFBox library, we have converted the PDF attachment file to an accessible PDDocument. Then we have started further processing of the PDF file page by page. We created PDF Rectangles which represents different areas or regions of the document for every page. Next we have created a PDFTextStripperByArea which extracts text from a specified region in the PDF. We have added these PDF Rectangles to this PDF Text Stripper for extraction of the text. PDFTextStripperByArea class also has been used to format the converted PDF text for ease of the further processing of the data. In this aspect we have added proper new line characters into the text and added tabs after each word. This format will help in extracting important information from the Text. After setting up this PDF Text Stripper with PDF Rectangles, New Line Character and word separators, we have used a method called GetTextForRegion which converts the whole file and returns the text version of the PDF. This is an efficient procedure and it has been found that this PDF text extraction by regions is useful for most of the PDFs.

Table Format: Often PDF files contain tabular data in it. We can use the same procedure that we have used for Normal Text using PDFBox for PDFs with Tables as well. However it will not keep the tabular data in a well-structured format and this will create obstacles in further processing of that file content. Hence keeping that problem in mind, we have used a separate procedure using PDFReader library to convert PDF with Tables to text format. It first converts the whole PDF file into HTML format. Then it reads the HTML formatted data by TAGs. If an <HTML> TAG is obtained a table in text is created which is being closed on encountering a

</HTML> tag. In between if it finds a <TR> tag which implies a TABLE ROW, it adds a new line character to the text. When a <TH> or <TD> tag is retrieved which respectively denotes the tag representation for a TABLE HEADER or TABLE DATA then it adds tabs to separate them from each other in the text. Inside a <TH> and </TH> or <TD> and </TD> the procedure looks for paragraph tags <P> and </P> because paragraphs contains the main content of a cell. Whatever the system gets inside a pair of paragraph tags, it adds that to the text. This text becomes the final converted form of the PDF.

We have used a predefined format to represent a table in plain text after conversion from PDF to the text. This specific format helps in further processing. For example, we can place new line characters after every row of the Table and can put tab characters after each column in a row.

3.2.3.2. EXCEL to Text Conversion

Excel attachments have been converted into Text using Java Excel API. We have created Work Book for each excel file, extracted the text and saved them in a String. First we start reading the Excel file from the top. We read the file on a line by line basis and visit each and every cell inside that row. We add the cell contents to the text file in the sequence that they have been visited. Whenever we go to a new line in the Excel sheet, we add a new line character to the text and when we go from one cell to another we add a tab in the text. We use this specific format to represent a table inside the text to make further processing of that text easier.

3.2.3. Email Body Extraction and Format Conversion

Our email client receives the emails, extracts the main body part and makes it available for further processing. Most of the emails are of HTML format. Hence the email client converts the HTML into plain text format regardless of the Tables and other High level HTML contents. Our work of text parsing for Information Extraction starts after that point.

3.3. Information Extraction

After we have retrieved both the email attachment and email body in their respective text formats, we begin processing those using different rules and algorithms to extract important information. We have observed different format of data representation in text like Tabular data, Natural Language and unformatted data. At the same time we had some other challenges which are discussed below:

3.3.1. Challenges

3.3.1.1. Date Format

In emails there are different date formats. People are conversant with different date formats while writing emails. Some of the commonly used date formats are represented as "Month-Date-Year" and "Date-Month-Year". We have discovered many possible date formats after a careful survey of many emails which are described in following table.

In Table 2 we have presented different possible date formats that can be found from emails. These formats have been captured after consulting numerous emails from different company datasets. During email extraction in our program we have examined all these formats to determine dates. We have converted all the dates from String to DateTime format using different rules. As a result of parsing different emails, we obtain date in one specific format. We are saving that format in the database. There are multiple advantages of converting all the dates into one single format. A company database can be defined in a specific DateTime format and as a result of this processing a perfect alignment can be achieved between the DateTime format of the user emails and that of the company database. Moreover a company might require all the emails to follow a single format for the purpose of further processing of the emails information in which case our processing comes really handy.

Table 2: Different Possible Date Formats for Emails

Date Format	Example
Date-Month-Year	21-10-2013
Month-Date-Year	09-23-2014
Date/Month/Year	21/10/2013
Month/Date/Year	09/23/2014
Date.Month.Year	21.10.2013
Month.Date.Year	09.23.2014
Date Month,Year	21st January, 2013
Date Month,Year	20th Jan, 2014
Month Date, Year	Jan 20, 2015
Month Date, Year	January 20th, 2013

3.3.1.2. City and States

In the emails that an organization receives, often cities and states are clearly mentioned. For example we have taken a dataset of a freight company. They receive emails about truck transportation requests. Those emails contain information about the source city and the destination city. Based on the requirements of the organizations, it is required to find out the cities and states. In our algorithm we have used a Dictionary matching algorithm to find out all mentioned states in the email by matching them with a List of states. Here we have worked with only USA and Canada states as the company has operational units only in USA and Canada. We have prepared a list of all states in USA and Canada. Based on the organization and its requirements, we can change the Dictionary entries for the states and we can search them in the emails. Once we achieve a match with any of the dictionary entries, we tag them as states. Later as described in the following section, we discriminate between source and destination locations and mark those states as source state or destination state.

3.3.1.3. Source Destination Discrimination

In our dataset of emails, we have observed emails where companies request for trucks to send a load from a source to a destination location. Using Named Entity Recognition, we were able to tag locations in the text. However it was a challenge to know which of them are destination and which are source. So we have implemented a set of rules using some keywords to discriminate between them. For example: there is a text like

"I have a shipment ready for delivery from Minneapolis, MN to Seattle, WA"

Here Minneapolis, MN is the source and Seattle, WA is the destination. We can find out that Minneapolis is the source city because it is followed by the keyword "from" and Seattle is the Destination city because it is followed by the keyword "to". So here we have jot down a bunch of these kind of keywords that can be helpful to find out the Source and Destination locations. Now after using Named Entity Recognition, as locations are marked, we are using Dictionary matching for those keywords and finding out the Source city, state and Destination city, state.

3.3.2. Table Data Processing

In different organization, we have encountered many emails that contains tabular data. Like a Pdf file can contain information which is organized in a table format or we can encounter excel file. Now as mentioned earlier, we have pre-processed all of these files and has generated text files. In text files for tabular data representation, we have created a format. We have given a special character after each new line i.e. starting of a new row and another special character after each entry i.e. each column entry. For example: bellow Table 3 represents a table with some student information.

Table 3: Example Table Data for Processing

No.	Student Name	Student Address	Student Phone No
01.	Nick	St. Paul	(780) 432-2839
02.	Brad	Dallas	(432) 120-2390

Now if we will convert this table into our previous described text format, it will become like following

No. <col> Student Name <col> Student Address <col> Student Phone No <row> 01. <col> Nick <col> St. Paul <col> (780) 432-2839 <row> 02. <col> Brad <col> Dallas <col> (432) 120-2390 <END>

Now when we are processing, we are reading the whole text and based on those special characters, we are getting those entries for the table. In tabular data processing the first main job is to find out the Table header information like here in this example No, Student Name, Student Address, Student Phone No. Here in our algorithm, we have created a list of possible table headers for our expected data entries that we need to extract for a specific organization. To extract those table headers first. We start reading the whole text and look for a row with some header matching. After we get that header row. We start reading other rows and enter those column values under their respected column header. So, for the above example, when we find the Header row, we read second row and put 01 under No, put Nick under Student Name etc. We continue this work until we get the end of the file or we get another Table header, means starting of another table. We use a Map to map Table headers to our required information name. So here we want Name of the Students so, we are looking for Table column name like “Student Name”, “Name”, “Student First Name”, “Student Last Name” etc. Based on those matching we extract and save table entries.

Algorithm: TableTextExtractor (String EmailText)

This function takes an Email Text which is was generated from Table content and parses it. Then it extracts all important data from that content.

```
1.  START
2.  String[] $Line_i$ set to <- Split the EmailText based on New Line character
3.  LineCount set to number of lines in the $Line_i$
4.  for all the line-s in $Line_i$ do
5.      String[] $Word_j$ set <- Split the $Line_i$ based on Tab character
6.      for all the word-s in $Word_i$ for line do
7.          Match word with Dictionary of Keys Table Headers
8.          if Match found do
9.              Map that Column of the word to that Category (from the Dictionary of Keys)
10.             Save them as TABLE HEADERs with their column numbers
11.         end if
12.     end for
13. end for
14. Now we have the Table Headers
15. for all the line-s in $Line_i$ after the TABLE HEADER do
```

```
16.     Create an *EmailEntry* for *line*
17.     for all the *word*-s in $Word_i$ for *line* do
18.         Map the *word* to the TABLE HEADER
19.         if Match found do
20.             Enter the *word* to the EmailEntry with its corresponding TABLE HEADER
21.         end if
22.     end for
23.     Add the *EmailEntry* to a EmailEntry List
24. End For
25. Return the EmailEntry List
26. END
```

3.3.3. Named Entity Recognition

Most of the emails that we encounter are written in Natural Language, so no doubt this is the most important and challenging part of the project. Here we have used Stanford's Named Entity Recognizer algorithm and library function [21]. As discussed earlier, in this algorithm they have used non-local information in information extraction using Gibbs sampling.

We have used different annotators like "tokenizer", "ssplit", "parts of speech", "lemma", "Named Entity" to annotate the email text. Now we have gone through all the sentence and tokenized them, then we have used Text annotation to get the words, Parts of speech annotation to get Parts of speech and Named Entity Tag Annotation to get Named entities. We have used some training dataset to train the NER Model. Here the NER model is trained with multiple datasets. We have used Stanford NER's default trained model. This model is trained with

CoNLL – 2002 and CoNLL -2003 (British newswire) and it supports Multiple languages like Spanish, Dutch, English and German. It contains 4 named entities: Person, Location, Organization and Misc.

MUC -6 and MUC -7 (American newswire) and it supports 7 named entities: Person, Location, Organization, Time, Date, Percent, and Money.

ACE which supports 5 named entities: Location, Organization, Person, FAC, GPE.

This NER model gives pretty much good result in our project.

Now we have used all of those annotated words to figure out our required information from the text. In our project we have taken a data set for Truck and load Assignment Company.

Table 4: Email Entry Fields for Extraction

Field	Type
Package Weight	Double
Package type	String
Origin City	String Array
Destination City	String Array
Origin State	String Array
Destination State	String Array
Available Date Time	Date Time
Delivery Date Time	Date Time
Required Trailer Type	String
Description	String
Price	Double

We have used Date Time, Location, Person, Money Named entities to get required information for the project. In our project we are trying to find out following information listed in Table 4

After we have tagged all the locations, we have used dictionary matching to find out the States and then used some rules to determine the source and destination from them. We have described earlier about the rules that we have used to find out destination and source. Now comes Date Time; as discussed earlier we have used many techniques to convert any format of date to a standard Date Time format to record that for father processing in the system. We often use words like today, tomorrow, Day after today, day after tomorrow or even we use like coming Monday or last Saturday. In this project we have used multiple rules to figure out the actual date for those days and has entered them into our database for farther processing. We have created conditions for keywords like “today”, “tomorrow”, “day after tomorrow” etc. and whenever we find them we calculate their date from today’s system date. We have created a Dictionary of Days (Sunday, Monday,..) and assigned a number to each of these days. Now whenever we get one of this Dictionary entry in the email text, we consult the Dictionary to get the corresponding day number for that. Now we take the current date from system clock and do some calculation to find out the date of that day which is coming. For example today’s date is 10/16/2014 and day is Thursday. I received an email today and they are pointing out on coming Monday. Now we use the Dictionary to find out the day number for today =4 and Monday =1. Then we calculate the day difference from today to Monday using that day number = -3. Now when the difference goes bellow 0 then add 7 to that as total number of days in a week is 7. So now the difference becomes 4. Now we add that difference to today’s date and it gives the date of coming Monday as 10/20/2014 (16 + 4).

3.3.4. Dictionary Matching

We have used dictionary matching in our algorithm a lot. We have created multiple dictionaries in the project like State Dictionary, Keywords Dictionary etc. Here I should mention that though we have NER implemented in our project, but it does not detect all the items correctly. For example some time, NER does not detect States correctly or sometimes it marks them incorrectly. So it is an easier solution to use a dictionary matching for those fields and because of that we have incorporated dictionary matching here. For dictionary matching sometimes we have used normal string matching and sometimes we have used Lingpipe [36] library functions. In some occasion we have already tokenized sentences, so we have used those tokens to match with Dictionary entries using basic java string matching.

For example we have extracted States from the email text using Dictionary matching with normal string matching in our project

Otherwise we have used LingPipe dictionary chunker to find out matches with the text. For example when we started processing Email body in our project, for dataset of Truck and Load allocation, we first search for the kind of trailer they are asking. So, here before we used Stanford NER on the text, we used LingPipe Dictionary chunker to find out the trailer type.

Algorithm: DictionaryChunk (String text, Dictionary dictionary)

This algorithm finds whether any of the Dictionary word exists in the text or not. If any exists then it finds and returns.

```
1. START
2. Prepare the Chunker from PipeLine
3. DictionaryChucker <- Create ExactDictionaryChunker using dictionary and
   IndoEuropeanTokenizer
```

```
4.  String output set to null
5.  Chunking <- Chunker.chunk (text)
6.  ChunkSet <- get ChunckingSet from Chunking
7.  for each $chunk_i$ in ChunkSet do
8.      int CHUNKSTART set to $chunk_i$.Start
9.      Int CHUNKEND set to $chunk_i$ .End
10.     Output <- text . Substring (CHUNKSTART, CHUNKEND)
11. end for
12. Return Output
13. END
```

3.5. Storage

After processing all these data from the emails we are now we can save the extracted important information into a repository for company's further work. In our project we have created individual XML files to save featured data for each individual emails that we have read from the email client. For our project we have worked on a dataset of Truck and Load allocation. So for emails asking for a truck we have extracted information about the kind of truck they are looking for and other relative information then we have saved them in an XML file that can be utilized to allocate a truck for that client.

CHAPTER 4. EXPERIMENT AND RESULT

4.1. Dataset

For our experiment we have taken a dataset of a company named ValleyExpress who allocates trucks and loads. We have taken 231 of emails which contains Natural language email body content, PDF or Excel files as attachments. All these emails have information regarding the source and destination locations for the load transfer, dates and times for the transfer etc. We are running our application on this set of emails.

4.2. Experiment

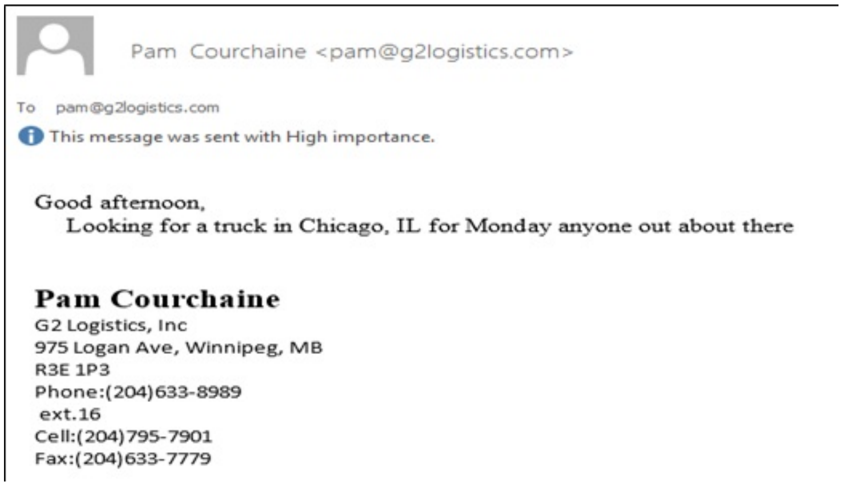
I am using an Email Client which receives all the emails come to the Company and gives my program following information

Senders email address, Emails Subject, Email body in text format and downloaded attachments. Emails body text comes in a .txt file and Email Client sends the link to the system. My system starts working from there; it reads all those files from the link and starts processing.

4.3. Result

In the Figure 7 we have shown a sample email that we have used in our system and has processed in the category of Natural Language for email body.

Sample email:



Pam Courchaine <pam@g2logistics.com>

To pam@g2logistics.com

This message was sent with High importance.

Good afternoon,
Looking for a truck in Chicago, IL for Monday anyone out about there

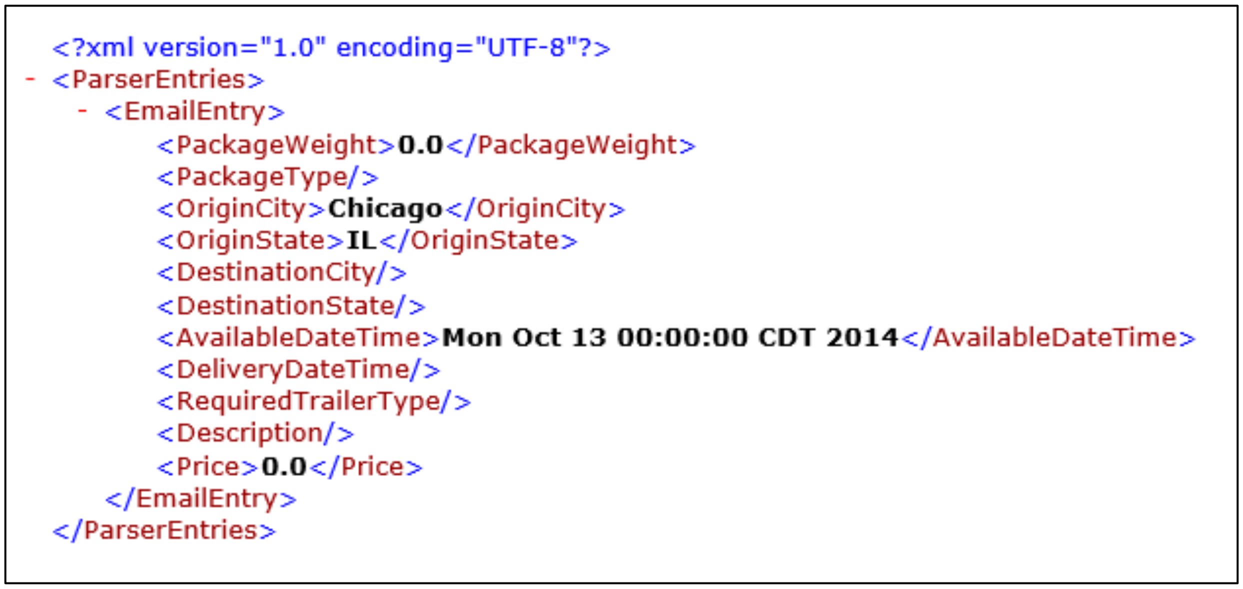
Pam Courchaine
G2 Logistics, Inc
975 Logan Ave, Winnipeg, MB
R3E 1P3
Phone:(204)633-8989
ext.16
Cell:(204)795-7901
Fax:(204)633-7779

Figure 7: Sample Email Body as Natural Language

This email does not have any subject and does not have any attachments associated with it. I have applied this to my system and got the following result shown in Figure 8 bellow.

Output:

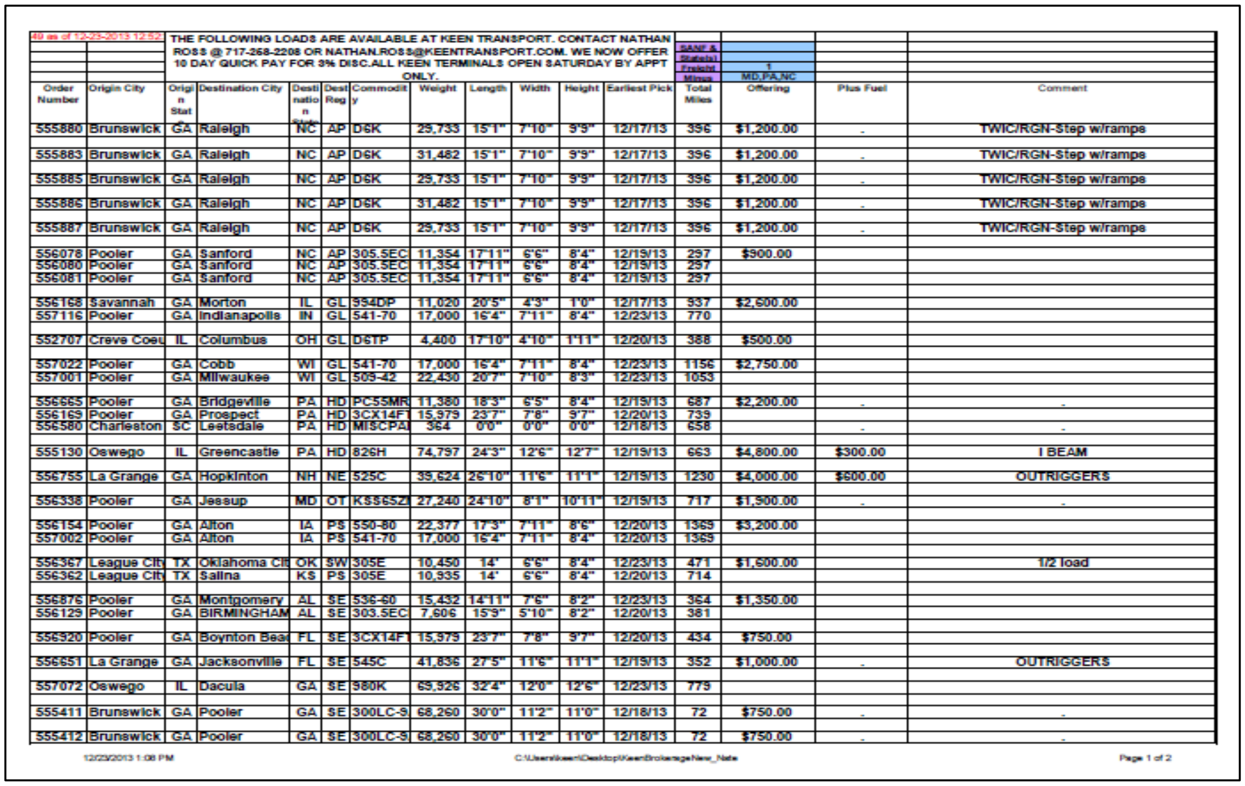
Kind of load is unknown here.



```
  <?xml version="1.0" encoding="UTF-8"?>
- <ParserEntries>
  - <EmailEntry>
        <PackageWeight>0.0</PackageWeight>
        <PackageType/>
        <OriginCity>Chicago</OriginCity>
        <OriginState>IL</OriginState>
        <DestinationCity/>
        <DestinationState/>
        <AvailableDateTime>Mon Oct 13 00:00:00 CDT 2014</AvailableDateTime>
        <DeliveryDateTime/>
        <RequiredTrailerType/>
        <Description/>
        <Price>0.0</Price>
    </EmailEntry>
  </ParserEntries>
```

Figure 8: Output for an Email with Email Body as Natural Language

Now we have presented an email which has a PDF attachment which gives all the details about the load transfer as shown in Figure 9



49 as of 12-23-2013 12:52

THE FOLLOWING LOADS ARE AVAILABLE AT KEEN TRANSPORT. CONTACT NATHAN ROSS @ 717-268-2208 OR NATHAN.ROSS@KEENTRANSPORT.COM. WE NOW OFFER 10 DAY QUICK PAY FOR 3% DISC.ALL KEEN TERMINALS OPEN SATURDAY BY APPT ONLY.

SANF &	
State(s)	
Freight	1
Miles	MD,PA,NC

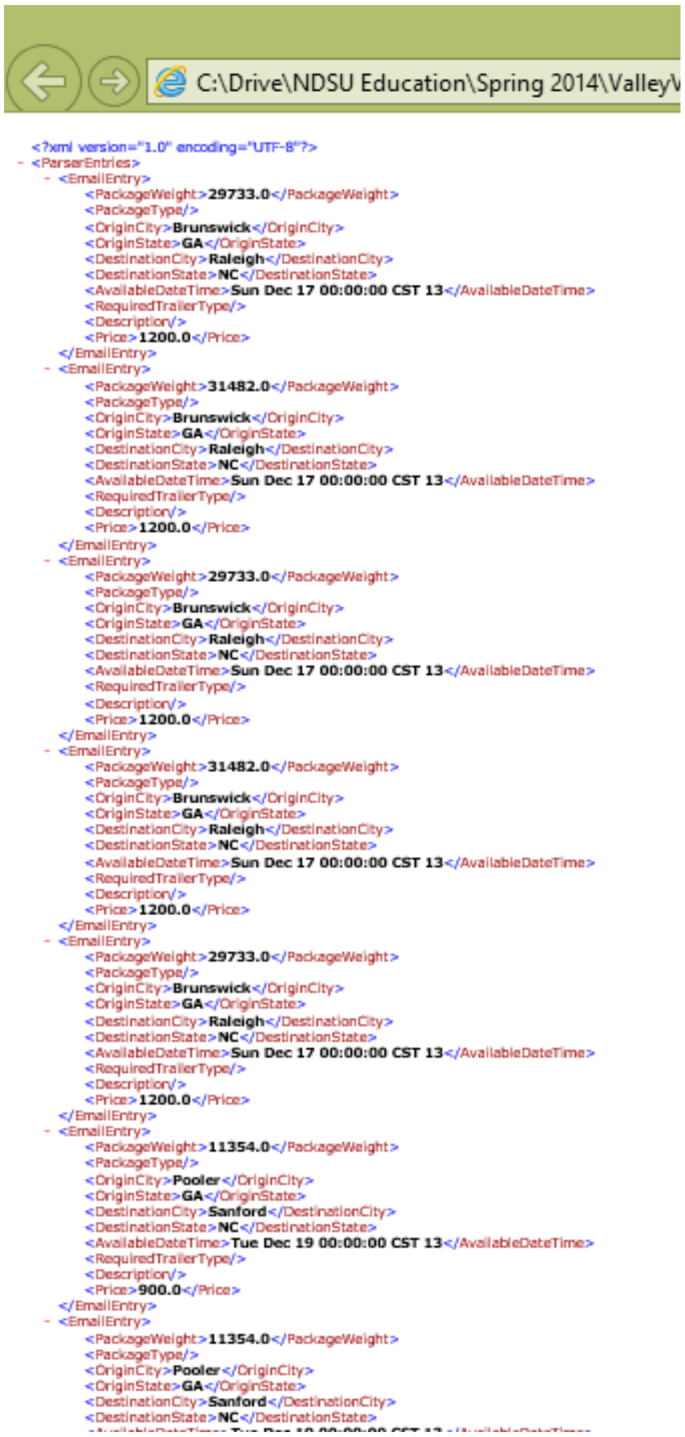
Order Number	Origin City	Origin Stat	Destination City	Destination State	Dest Reg	Commodity	Weight	Length	Width	Height	Earliest Pick	Total Miles	Offering	Plus Fuel	Comment
555880	Brunswick	GA	Raleigh	NC	AP	D6K	29,733	15'1"	7'10"	9'9"	12/17/13	396	$1,200.00	.	TWIC/RGN-Step w/ramps
555883	Brunswick	GA	Raleigh	NC	AP	D6K	31,482	15'1"	7'10"	9'9"	12/17/13	396	$1,200.00	.	TWIC/RGN-Step w/ramps
555885	Brunswick	GA	Raleigh	NC	AP	D6K	29,733	15'1"	7'10"	9'9"	12/17/13	396	$1,200.00	.	TWIC/RGN-Step w/ramps
555886	Brunswick	GA	Raleigh	NC	AP	D6K	31,482	15'1"	7'10"	9'9"	12/17/13	396	$1,200.00	.	TWIC/RGN-Step w/ramps
555887	Brunswick	GA	Raleigh	NC	AP	D6K	29,733	15'1"	7'10"	9'9"	12/17/13	396	$1,200.00	.	TWIC/RGN-Step w/ramps
556078	Pooler	GA	Sanford	NC	AP	305.5ECI	11,354	17'11"	6'6"	8'4"	12/19/13	297	$900.00		
556080	Pooler	GA	Sanford	NC	AP	305.5ECI	11,354	17'11"	6'6"	8'4"	12/19/13	297			
556081	Pooler	GA	Sanford	NC	AP	305.5ECI	11,354	17'11"	6'6"	8'4"	12/19/13	297			
556168	Savannah	GA	Morton	IL	GL	994DP	11,020	20'5"	4'3"	1'0"	12/17/13	937	$2,600.00		
557116	Pooler	GA	Indianapolis	IN	GL	541-70	17,000	16'4"	7'11"	8'4"	12/23/13	770			
552707	Creve Coeu	IL	Columbus	OH	GL	D6TP	4,400	17'10"	4'10"	1'11"	12/20/13	388	$500.00		
557022	Pooler	GA	Cobb	WI	GL	541-70	17,000	16'4"	7'11"	8'4"	12/23/13	1156	$2,750.00		
557001	Pooler	GA	Milwaukee	WI	GL	509-42	22,430	20'7"	7'10"	8'3"	12/23/13	1053			
556665	Pooler	GA	Bridgeville	PA	HD	PC55MR	11,380	18'3"	6'5"	8'4"	12/19/13	687	$2,200.00	.	.
556169	Pooler	GA	Prospect	PA	HD	3CX14FT	15,979	23'7"	7'8"	9'7"	12/20/13	739			
556580	Charleston	SC	Leetsdale	PA	HD	MISCPAI	364	0'0"	0'0"	0'0"	12/18/13	658		.	.
555130	Oswego	IL	Greencastle	PA	HD	826H	74,797	24'3"	12'6"	12'7"	12/19/13	663	$4,800.00	$300.00	I BEAM
556755	La Grange	GA	Hopkinton	NH	NE	525C	39,624	26'10"	11'6"	11'1"	12/19/13	1230	$4,000.00	$600.00	OUTRIGGERS
556338	Pooler	GA	Jessup	MD	OT	KSS65ZI	27,240	24'10"	8'1"	10'11"	12/19/13	717	$1,900.00	.	.
556154	Pooler	GA	Alton	IA	PS	550-80	22,377	17'3"	7'11"	8'6"	12/20/13	1369	$3,200.00		
557002	Pooler	GA	Alton	IA	PS	541-70	17,000	16'4"	7'11"	8'4"	12/20/13	1369			
556367	League City	TX	Oklahoma Cit	OK	SW	305E	10,450	14'	6'6"	8'4"	12/23/13	471	$1,600.00		1/2 load
556362	League City	TX	Salina	KS	PS	305E	10,935	14'	6'6"	8'4"	12/20/13	714			
556876	Pooler	GA	Montgomery	AL	SE	536-60	15,432	14'11"	7'6"	8'2"	12/23/13	364	$1,350.00		
556129	Pooler	GA	BIRMINGHAM	AL	SE	303.5ECI	7,606	15'9"	5'10"	8'2"	12/20/13	381			
556920	Pooler	GA	Boynton Bea	FL	SE	3CX14FT	15,979	23'7"	7'8"	9'7"	12/20/13	434	$750.00		
556651	La Grange	GA	Jacksonville	FL	SE	545C	41,836	27'5"	11'6"	11'1"	12/19/13	352	$1,000.00	.	OUTRIGGERS
557072	Oswego	IL	Dacula	GA	SE	980K	69,926	32'4"	12'0"	12'6"	12/23/13	779			
555411	Brunswick	GA	Pooler	GA	SE	300LC-9	68,260	30'0"	11'2"	11'0"	12/18/13	72	$750.00	.	.
555412	Brunswick	GA	Pooler	GA	SE	300LC-9	68,260	30'0"	11'2"	11'0"	12/18/13	72	$750.00	.	.

12/23/2013 1:08 PM C:\Users\keen\Desktop\KeenBrokerageNew_Nate Page 1 of 2

Figure 9: Sample PDF Attachment with Table in It

Now we have extracted the attachment and fit into our system. Our system has converted the PDF into text with the Table first and then has processed the file to extract required fields of information. Output for the email with PDF attachment is shown in Figure 10.

We have used our system on 110 emails as Natural Language as email body, 54 emails with PDF with Text as attachment, 42 emails as PDF with tables as attachment and 25 emails as Excel files as attachment. We have counted the number of Desired Fields that appears in the email content and the number of fields that we have managed to capture. Based on the result found in the project, we have calculated Precision and Recall, which have been presented in the flowing Table 5.


C:\Drive\NDSU Education\Spring 2014\ValleyV

```
  <?xml version="1.0" encoding="UTF-8"?>
- <ParserEntries>
  - <EmailEntry>
        <PackageWeight>29733.0</PackageWeight>
        <PackageType/>
        <OriginCity>Brunswick</OriginCity>
        <OriginState>GA</OriginState>
        <DestinationCity>Raleigh</DestinationCity>
        <DestinationState>NC</DestinationState>
        <AvailableDateTime>Sun Dec 17 00:00:00 CST 13</AvailableDateTime>
        <RequiredTrailerType/>
        <Description/>
        <Price>1200.0</Price>
    </EmailEntry>
  - <EmailEntry>
        <PackageWeight>31482.0</PackageWeight>
        <PackageType/>
        <OriginCity>Brunswick</OriginCity>
        <OriginState>GA</OriginState>
        <DestinationCity>Raleigh</DestinationCity>
        <DestinationState>NC</DestinationState>
        <AvailableDateTime>Sun Dec 17 00:00:00 CST 13</AvailableDateTime>
        <RequiredTrailerType/>
        <Description/>
        <Price>1200.0</Price>
    </EmailEntry>
  - <EmailEntry>
        <PackageWeight>29733.0</PackageWeight>
        <PackageType/>
        <OriginCity>Brunswick</OriginCity>
        <OriginState>GA</OriginState>
        <DestinationCity>Raleigh</DestinationCity>
        <DestinationState>NC</DestinationState>
        <AvailableDateTime>Sun Dec 17 00:00:00 CST 13</AvailableDateTime>
        <RequiredTrailerType/>
        <Description/>
        <Price>1200.0</Price>
    </EmailEntry>
  - <EmailEntry>
        <PackageWeight>31482.0</PackageWeight>
        <PackageType/>
        <OriginCity>Brunswick</OriginCity>
        <OriginState>GA</OriginState>
        <DestinationCity>Raleigh</DestinationCity>
        <DestinationState>NC</DestinationState>
        <AvailableDateTime>Sun Dec 17 00:00:00 CST 13</AvailableDateTime>
        <RequiredTrailerType/>
        <Description/>
        <Price>1200.0</Price>
    </EmailEntry>
  - <EmailEntry>
        <PackageWeight>29733.0</PackageWeight>
        <PackageType/>
        <OriginCity>Brunswick</OriginCity>
        <OriginState>GA</OriginState>
        <DestinationCity>Raleigh</DestinationCity>
        <DestinationState>NC</DestinationState>
        <AvailableDateTime>Sun Dec 17 00:00:00 CST 13</AvailableDateTime>
        <RequiredTrailerType/>
        <Description/>
        <Price>1200.0</Price>
    </EmailEntry>
  - <EmailEntry>
        <PackageWeight>11354.0</PackageWeight>
        <PackageType/>
        <OriginCity>Pooler</OriginCity>
        <OriginState>GA</OriginState>
        <DestinationCity>Sanford</DestinationCity>
        <DestinationState>NC</DestinationState>
        <AvailableDateTime>Tue Dec 19 00:00:00 CST 13</AvailableDateTime>
        <RequiredTrailerType/>
        <Description/>
        <Price>900.0</Price>
    </EmailEntry>
  - <EmailEntry>
        <PackageWeight>11354.0</PackageWeight>
        <PackageType/>
        <OriginCity>Pooler</OriginCity>
        <OriginState>GA</OriginState>
        <DestinationCity>Sanford</DestinationCity>
        <DestinationState>NC</DestinationState>
```

Figure 10: Sample Output from PDF Table

Precision: Precision represents fraction of retrieved items which are relevant i.e. the number of correct results delivered divided by the number of all items retrieved.

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

Recall: Recall represents fraction of relevant items that has been retrieved i.e. number of correct results achieved divided by the number of correct results that were supposed to be returned. [38]

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

Table 5: Shows Precision and Recall for the Experiment

Type of Document	Avg. Number of Fields Present	Avg. Number of Correct Fields Retrieved	Avg. Number of Fields Retrieved	Precision	Recall
NATURAL LANGUAGE	5.24	4.19	5.02	83.4 %	79.9 %
PDF as TEXT	5.18	4.01	5.06	79.24 %	77.42 %
PDF as TABLE	6.45	5.29	6.81	77.7 %	82.1 %
EXCEL	6.23	5.11	6.72	76.04 %	82.03 %

4.4. Application Run Time:

Application running time varies upon whether we are doing batch processing or single email processing. As we have to load lots of Library functions, it takes some time to load all of them. However once they have been loaded, then processing emails does not take too much of time. As estimated for single email processing it might take 20 sec to process the content and return the result.

CHAPTER 5. CONCLUSION AND FUTURE WORK

5.1. Conclusion

Emails are really important in our daily personal life as well as in the industries. There are numerous examples of industries where emails are the only way of communication between them and their clients. Service based companies provide services through emails. So it is really necessary to get all the information from the emails clearly and correctly.

In our project we have implemented a system of encountering these emails; especially for industries. We have implemented rules and algorithms in our system which deals with all these emails received in the email client. Our system pre-process the emails, finds out sender information, subjects, separates all the attachments and then process all of them including the email body to extract necessary data from it. It handles lots of other difficulties about email formats and text formats to produce best possible result.

We have seen, if the emails have similar structures or has been generated from a same template, the system works very well. Whereas with vastly different formats in the emails, the performance of the system goes down a little bit.

5.2. Future Work

5.2.1. Email Signature Removal

Often emails are associated with signatures based on the industry we are dealing with. Sometimes email signature information create confusion in the information data extraction. Sometimes signature information gets extracted as part of the service information and that is not desired. So it is significant to remove the email signature before we start information extraction from the email. This will produce better result.

5.2.2. Encountering Unknown Formats

Here in our project we have encountered and has implemented the logic to deal with the emails that have Natural Language in the body or Tabular information in the body. However sometimes emails do not contain information is a specific format, instead of that information is little bit messy. We have not encountered them here. This can be done in future for better performance and better result with many emails.

5.2.3. Images and Other Icons Handling

Many emails contain images or other icons. As part of important information extraction, these icons and images are often not required. These unnecessary icons can be removed during the pre-processing of the emails. This will keep the information extraction process simple and easy.

In some aspect, images or icons can appear into the required information that the organization wants to extract. Then processing and extracting those icons, images and storing them will be required.

5.2.4. Incorrect Spelling

Emails often contain incorrect spellings and it creates problem to extract those information correctly. Here we can use a spelling correcting algorithm to correct all those incorrect spelled words before we start the process of extraction of information.

5.2.5. Microsoft Word Files as Attachments

We have implemented extraction of pdf and excel file attachment from the emails and we have process of converting them to a normal text format to process them for extracting information. Microsoft word files or some other format files can be implemented as part of this system as well.

REFERENCES

[1] Tang, G., Pei, J., & Luk, W. S. (2013). Email mining: tasks, common techniques, and tools. *Knowledge and Information Systems*, 1-31.

[2] Kok, S., & Yih, W. T. (2009). Extracting product information from email receipts using markov logic. In *Proceedings of the Sixth Conference on Email and Anti-Spam, Mountain View, California, USA*.

[3] “Pew Internet Report: Online Activities 2010”. Pew Research Center. May 2010. Web. Aug 2014. <http://tinyurl.com/ pewOnline10>

[4] “Jones, J.: Gallup: Almost All E-Mail Users Say Internet, E-Mail Have Made Lives Better.” Gallup. July 2001. Web. Aug 2014. <http://tinyurl.com/Gallup01>

[5] “The Radicati Group, Inc.: Email Statistics Report, 2010.” Editor: Sara Radicati. The Radicati Group Inc. 2010. Web. Aug 2014. <http://tinyurl.com/RadicatiEmail10>

[6] “Taming the Growth of Email – An ROI Analysis (White Paper).” HP, The Radicati Group, Inc. Mar 2005. Web. Sept 2014. <http://tinyurl.com/RadicatiEmail05>

[7] “80 % of Users Prefer E-Mail as Business Communication Tool.” META Group Inc. 2003. Web. Sept 2014. <http://tinyurl.com/MetaEmail03>

[8] “Networked Workers. PewInternet report” Madden, M.—Jones, S. Pew Research Center. Sept 24, 2008. Web. Sept 2014. <http://tinyurl.com/pewNetWrks08>

[9] Laclavík, Michal, et al. "Email analysis and information extraction for enterprise benefit." *Computing and informatics* 30.1 (2012): 57-87.

[10] Whittaker, Steve, and Candace Sidner. "Email overload: exploring personal information management of email." *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1996.

[11] Fisher, D.—Brush, A. J.—Gleave, E.—Smith, M.A.: Revisiting Whit-taker&Sidner's "Email Overload" Ten Years Later. In CSCW2006, New York ACM Press 2006.

[12] Corbató, F. J., Merwin-Daggett, M., & Daley, R. C. (1962, May). An experimental time-sharing system. In *Proceedings of the May 1-3, 1962, spring joint computer conference* (pp. 335-344). ACM.

[13] "Natural Language Processing." Wikipedia, the free encyclopedia. Wikimedia Foundation, Inc. 26 October 2014. Web. 27 October 2014. <http://en.wikipedia.org/wiki/Natural_language_processing>

[15] "Named Entity Recognition." Wikipedia, the free encyclopedia. Wikimedia Foundation, Inc. 26 October 2014. Web. 27 October 2014. <http://en.wikipedia.org/wiki/Named-entity_recognition>

[16] Zhou, GuoDong, and Jian Su. "Named entity recognition using an HMM-based chunk tagger." *proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2002.

[17] Cunningham, H.—Maynard, D.—Bontcheva, K.—Tablan, V.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL '02), Philadelphia.

[18] “Se-mantic Annotation, Indexing, and Retrieval. Elsevier’s Journal of Web Semantics, Vol. 2, 2005, No. 1.” Kiryakov, A.—Popov, B.—Terziev, I.—Manov, D.—Ognyanoff, D. Ontotext. 2005. Web. Sept 2014. <http://www.ontotext.com/kim/semanticannotation.html>

[19] Cimiano, P.—Ladwig, G.—Staab, S.: Gimme’ the Context: Context-Driven Automatic Semantic Annotation With C-Pankow. In WWW’05: Proceedings of the 14th international conference on World Wide Web, New York, NY, USA. ACM Press. ISBN 1-59593-046-9, 2005, pp. 332–341.

[20] Laclavík, Michal, et al. "Email analysis and information extraction for enterprise benefit." *Computing and informatics* 30.1 (2012): 57-87.

[21] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non local Information into Information Extraction Systems by Gibbs Sampling. Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005), pp. 363 370. <http://nlp.stanford.edu/~manning/papers/gibbscrf3.pdf>

[23] “Gazetteer.” Wikipedia, the free encyclopedia. Wikimedia Foundation, Inc. 01 August 2014. Web. 20 October 2014. < http://en.wikipedia.org/wiki/Gazetteer>

[24] Aho, Alfred V, Corasick, Margaret J. (June 1975). "Efficient string matching: An aid to bibliographic search". Communications of the ACM 18 (6): 333–340. doi:10.1145/360825.360855.

[25] “Bio sequence Algorithms, spring 2005 Lecture 4: Set Matching and Aho-Corasick Algorithm.” Kilpelainen, Pekka. 2005. Sept 2014. <http://www.cs.uku.fi/~kilpelai/BSA05/lectures/ slides04.pdf >

[26] M. Richardson and P. Domingos. Markov logic networks. Machine Learning, 62:107–136, 2006.

[27] Kok, Stanley, and Wen-tau Yih. "Extracting product information from email receipts using markov logic." *Proceedings of the Sixth Conference on Email and Anti-Spam, Mountain View, California, USA*. 2009.

[28] Boufaden, Narjes, et al. "PEEP-An Information Extraction base approach for Privacy Protection in Email." *CEAS*. 2005.

[29] “Apache PDFBox – A Java Pdf Library.” The Apache Software Foundation. 2014. Web. Sept2014. <https://pdfbox.apache.org/>

[30]_Wasi, Shaukat, et al. "Event Information Extraction System (EIEE): FSM vs HMM."

[31] Saleem, Ozair, Latif, Seemab. “Information Extraction from Research Papers by Data Integration and Data Validation from Multiple Header Extraction Sources.” WCECS 2012, October 24-26, 2012, San Francisco, USA. <http://www.iaeng.org/publication/WCECS2012/WCECS2012_pp215-219.pdf >

[32] Chiticariu, Laura, et al. "SystemT: an algebraic approach to declarative information extraction." *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010.

[33] Almgren, Magnus, and Jenny Berglund. "Information extraction of Seminar information." *CS224N: Final Project* (2000): 1-12.

[34] Black, Julie A., and Nisheeth Ranjan. "Automated event extraction from email."*Final Report of CS224N/Ling237 Course in Stanford: http://nlp. stanford. edu/courses/cs224n/2004/, Spring* (2004).

[35] “Apache PDFBox 1.8.6 API.” The Apache Software Foundation. 2014. Web. Oct 2014. http://pdfbox.apache.org/docs/18.6/javadocs

[36] Cimiano, Philipp, Günter Ladwig, and Steffen Staab. "Gimme'the context: context-driven automatic semantic annotation with C-PANKOW." *Proceedings of the 14th international conference on World Wide Web*. ACM, 2005.

[37] Etzioni, O.—Cafarella, M.—Downey, D.—Kok, S.—Popescu, A.—Shaked, T.—Soderland, S.—Weld, D.—Yates, A.: Web-Scale Information Extraction in Knowitall (Preliminary Results). In WWW’04, 2004, pp. 100–110, http://doi.acm.org/10.1145/988672.988687.

[38] “Precision and Recall.” Wikipedia, the free encyclopedia. Wikimedia Foundation, Inc. 29 October 2014. Web. 31 October 2014. < http://en.wikipedia.org/wiki/Precision_and_recall>