International Journal For Technological Research In Engineering
Volume 5, Issue 11, July-2018 *ISSN (Online): 2347 - 4718*

INFORMATION RETRIEVAL FROM UNSTRUCTURED DATA

Manish Srivastava[1], Mr. Wasif Khan[2]
[2]Assistant Professor, [1,2]Department of Computer Science and Engineering,
S.R.Institute of Management& Technology, Lucknow

Abstract: the Internet has an enormous amount of data in the unstructured form, unstructured data most often is written in natural languages like English or Hindi and does not have a predefined data model, and it can't be mapped directly to relational databases. This data exists in form of emails, Google searches, eBooks, social media content, blogs etc., in short, it's the written word and is everywhere in our lives. Unstructured data also exists in form of audio, images, and video in addition to the text format, the current best approach to make all this non text-data searchable is to add appropriate keywords, labels, and text descriptions or convert audio to text using speech recognition so that it could be used as text, so it does not matter how much structure is added to this unstructured data, it all comes back to text for us to understand our content. Once we have unstructured data in text form, the next challenge is to extract high-quality (relevant, novel, and interesting) and useful information from it. One of the foremost challenges in retrieving information from the unstructured text written in natural languages is to transform it into structured attribute-based instances, for this purpose various natural language processing techniques such as encoding/normalization, tokenization, removing whitespaces/stop-words, parts of speech tagging, lemmatization or stemming, named entity recognition and sentence detection are applied. After text processing, various different algorithms may be applied for finding different types of information as per information need.
Keywords: Email spam detection, Machine Learning, Naïve Bayes Classification, Natural Language Processing, Text extraction, , Text Mining.

I. INTRODUCTION

With the emergence of social media, blogs, instant messaging, emails and various other technologies like IOT we are living in an era of data deluge, user-generated information is increasing at a phenomenal rate every second. A report by International Data Corporation (IDC) predicts the expected growth of digital universe by a factor of 300 (from 130 Exabyte to 40000 Exabyte) by the year 2020[1]. Maximum data on the internet is in unstructured form and cannot readily be indexed or mapped onto standard database fields; hence techniques are needed for processing and analyzing data for finding relevant information. Apart from text, data is present in various forms like audio, video, images etc. this non-text data could be made findable by adding text descriptions, keyword tags so that this unstructured data could be treated as text, thus, in the end, it's the text that is needed for understanding the content. Maximum data on the internet is in unstructured form and cannot readily be indexed or mapped onto standard database fields; hence techniques are needed for processing and analyzing data for finding relevant information. Apart from text, data is present in various forms like audio, video, images etc. this non-text data could be made findable by adding text descriptions, keyword tags so that this unstructured data could be treated as text, thus, in the end, it's the text that is needed for understanding the content. Unstructured data is mostly in text form, but it may also contain data like numbers, facts, quotes, protagonist and/or dates. This results in ambiguities and irregularities making it hard to understand using our conventional computer program when compared with relational databases. Managing unstructured data is considered as one of the most difficult issues in the computer science discipline, the main reason is that the tools, techniques, and algorithms that have been proven successful in converting "structured data" into "business intelligence" and "actionable information", don't work when applied to unstructured data.

II. ANALYZING UNSTRUCTURED DATA

A. Challenges in Analyzing Unstructured Data

Though text contains data, it's important to remember that text is not data. To extract some meaningful and relevant information from the unstructured text we have to add "structuring" to mold it into a shape fit for analysis.

As most of the unstructured text data is in the form of natural languages like Hindi or English so first and foremost challenge is the interpretation of information between human versus machine. To a machine, information is just a sequence of bits but to a human, its sequence of words, sentences and has some meaning to it. Since humans interact in natural languages like Hindi, Chinese, English etc. hence "Natural Language Processing" a computer science field is used in analysis and synthesis of natural languages and speeches.

Text data has rules of syntax, grammar, and expression, resulting in similar content getting interpreted as having different meanings. 'Search Jack' is not the same as saying 'Search, Jack'. Interpretation is also domain/context-sensitive; 'Bank is crumbling' has a different meaning when we look at the context, 'Bank may refer to a financial institution' or to the 'Bank of a river'. The text could also acquire different meanings when used in media and entertainment or in say, medical research.

Human factors also play a role in working with text, there are dialect-specific or culture-specific nuances, sarcasm, and emotions that alter meaning that must be inferred from context than mere words. Different cultures, different languages, and different interpretations of the exact same writing further complicate analysis.

Unstructured data is not in just one place, often it's distributed across the Internet in form of various documents,

the problem is to search for best-suited documents which are related to the specific query or domain, and extract the needed information. Mostly related documents are not of the same type, they may be in form of HTML, TXT, XML, JSON and proprietary formats like Adobe PDF and Microsoft word.

B. Text Processing

For the given information need we require processing the unstructured text data, add some structure to it to make it fit for analysis. Text Processing tasks include following steps

- Acquiring the data: Data can come from a variety of different sources like web pages (HTML), XML/JSON, PDF, Microsoft-Word, and CSV (comma separated value) formats. Extracting text from proprietary formats like PDF, or MS-Word can be done using various open source libraries like PDFBOX and Apache-POI.
- Cleaning the data: Once text is extracted from acquired data the text might still appear in some other different format like XML (or some different proprietary format). The next step is extracting only the actual text and segmenting it into parts of the document, e.g., abstract, body, title, headline, and so on. Finally, encoding of extracted text is normalized to ensure that characters are presented the same way, for example, documents encoded in formats such as ASCII, ISO 8859-1, and Windows-1250 are transformed into Unicode encoding.
- Natural Language Processing: An application that analyses unstructured text needs the ability to understand natural language; it can achieve this by splitting raw strings up into individual/separate words and then need to interpret what role each of those words play in the sentence (part of speech) and how they relate with each other via things like phrases and clauses.
- Text Mining: Text mining or Text analytics, is defined as the process of inspecting large collections of written resources of various analysis methodologies; NLP or natural language processing is one of them.

Text mining process is depicted in the following figure

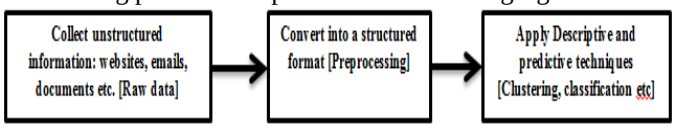

Figure 1: Text Mining

C. Collecting and Cleaning Data

1) Challenges in collecting data: Data can be collected from various sources like the Internet, databases, and disc storage. Sources of data may be different but real challenge appears (which is most often the case) when data is in different format like HTML, PDF, CSV, JSON, Microsoft Office (PowerPoint, MS-Word, Excel) etc. The first challenge appears when data exists in form of proprietary formats like Adobe PDF or MS-word etc., as a solution to this problem various open source libraries exist which help in extracting text from different proprietary formats such as PDFBox (extracting text from PDF files).

2) Cleaning Data: It is important to ensure that the text to be analyzed is in "correct format" before text analysis is complete. Data extracted from different formats may have Character encoding issues, it may have unnecessary whitespaces and we may need to remove stop words from the extracted text, stop words are the words like 'a', 'an', 'the' etc. that add very little meaning to the sentence and are often ignored by search engines.

Common text cleaning operations include:

- Removing unnecessary whitespaces.
- Removing non-text/special characters from the extracted text which may appear due to different file formats.
- Removing stop words.
- Converting text to lowercase.
- Finding and replacing text.
- Data imputation

D. Natural LanguageProcessing (NLP)

An application that analyses unstructured text needs the ability of understanding natural language; it can achieve this by splitting raw strings up into individual words and then interpret what role each of those words play in the sentence (part of speech) as well as how they relate to each other via things like phrases and clauses[1]. With this kind of information, it'll then be able take a query like "Who is Ram's father?" and dissect it to know that this query requires the answer to be a proper name (which consists of words tagged as nouns) and that it must appear in the same sentence as the words Ram and father (likely in that order). Following is the list of basic NLP tasks:

1) Tokenization: The very first step after extracting content from a file is breaking the content into small, usable chunks of text, called tokens. Tokens often represent single words, but tokens can be specific to an application. The most common approach to tokenizing English is splitting up a raw string based on the occurrence of whitespaces such as line breaks and spaces. Many techniques such as Case alterations, Stop word removal, and expansion can be applied at the token level to make text analysis easier. Expansion is the process of adding synonyms or expanding abbreviations and acronyms in a token stream which allows applications to handle alternative inputs from users (e.g., I.B.M or IBM).

2) Stemming: It is the task of reducing inflected/derived words to their word stem, root or base form—generally a written word form. Stemming is often used in text processing applications like search because users usually expect to find documents on "caterers" when searching for the word "catering", in other words, users may not necessarily need an exact match for their keywords. There are many different approaches to stemming, some are aggressive, and reduce words to the smallest root possible, whereas others are lighter, preferring to do basic things like removing 's' or 'ing' from words. Aggressive stemming usually leads to more results but lower quality, whereas light stemming can preserve some quality at the risk of missing some useful results. (Hull, 1995), has concluded that some form of stemming is almost always beneficial[2].

3) Lemmatization: Lemmatization is the task of determining a word's lemma based on its expected meaning. In contrast to stemming, lemmatization entirely depends on correctly recognizing the expected part of speech and meaning of a word in a sentence, as well as within the larger context surrounding that sentence, such as nearby sentences or an entire document.

4) Stemming vs. Lemmatization: The aim of both lemmatization and stemming is reducing inflexional forms of a word to a common base form, for example, the base word for 'cats' is 'cat'. However, the stemming and lemmatization differ in their essence. Stemming usually refers to the process of shortening a word by cutting off the end of words, for example, simple stemming of word 'studies' is 'studi'. Stemming achieves this goal correctly (most of the time) by removing derivational affixes. Lemmatization refers to doing things in a proper way, by using a dictionary and morphologically analyzing words (for example, "studies" becomes "study").

5) Parts of Speech Tagging: Identifying a word's part of speech (POS), such as whether it's a noun, verb, or adjective, is commonly used to enhance the quality of results in downstream processing. For instance, using part of speech can help determine important keywords in a document or to assist in searching for specific usages of a word (such as Can as a proper noun as in "The quantity contained in a can" versus can the verb, as in "You can do that"). There are many readily available, trainable part of speech taggers available as open source. In the proposed system OpenNLP Maximum Entropy Tagger is used, it uses statistics to figure out which part of speech is most likely for a word. The OpenNLP English POS tagger uses part of speech tags from the Penn Treebank Project [3], for labeling words in sentences with their part of speech.

6) Named Entity Recognition (NER): This NLP task is also called as entity identification and is used to identify People, places, and things in text. This helps in identifying the sentence's subject and often its object. Due to the importance of named entities (proper nouns), it's often useful when processing text to try to identify nouns and use them in our applications. Named entities concept in many cases is also used to identify temporal and numeric concepts like May 2018 or $60.35. Identifying people, organization, places, and other named entities allow us to capture what a text is about in an actionable way. For instance, with this information, we can provide more information about these entities, suggest other content which also features them or is related to them, and ultimately increase user engagement.

7) Sentence Boundary Disambiguation: Finding sentences is a task in natural language processing of deciding the boundary of a sentence i.e., where a sentence begins and ends, this problem is referred to as Sentence boundary disambiguation. It's typically used for downstream processes such as determining the meaning of a portion of text. Computing sentence boundaries can help reduce erroneous phrase matches as well as provide a means to identify structural relationships between words and phrases and sentences to other sentences. With these relationships, we can then attempt to find meaningful pieces of information in the text.

Common Techniques for finding sentences: Following is the simple approach of finding sentence boundary:

a) A period (.), question mark (?) or exclamation mark (!) ends a sentence.
b) If next token begins with the capital letter then the current token ends a sentence.
c) If the previous token is a list of abbreviations then it does not end the sentence.

This strategy gets about ninety-five percent of sentences correct [4].

E Text Mining

Text Mining is the process of automatically extracting high-quality information from text documents most often written in natural languages. Text mining has various applications; few of them are listed below:

- Classification/Categorization of text into specific domains.
- Text clustering for automatically organizing a set of documents.
- Sentiment analysis for identifying and extracting subjective information in documents.
- Entity extraction that is capable of identifying locations, people, organizations and other entities like the temporal entity from documents.
- Document summarization for automatically providing the most important features in the original document; and learning relations between named entities.

F. Natural Language Processing vs. Text Mining

Natural language processing (NLP) and text mining processes are usually employed together to extract relevant and meaningful information from the unstructured text which may be distributed across an enormous number of documents. They both work together with NLP acting as a component/module of Text mining processes.

Natural language processing (or NLP) performs a special kind of semantic analysis which essentially helps a machine in "reading" text. It's a part/component of text mining. NLP uses various methodologies for deciphering the ambiguities in human language, including the following: automatic summarization, tagging part-of-speech, disambiguation, extracting entities (NER) and relations, as well as understanding and recognizing natural language.

In text mining, relevant information in a text is discovered by converting the text into data that could be used for further analysis. Text mining achieves this by using many different analysis methodologies; NLP (natural language processing) is one of them.

G. Machine Learning

1)Machine learning is an area of Artificial Intelligence which gives a computer the capacity to automatically "learn" and progressively "improve performance" without any explicit programming.

2) Supervised vs. Unsupervised learning - In supervised learning, some external inputs or datasets are provided to the machine whereas in unsupervised learning no external inputs

or datasets are provided.

3) Clustering vs. Classification – Both classification and clustering algorithms are opposite sides of the same coin, both are used for document categorization.

Text classification is the application area which we have used for detecting email spam. Spam not only exists in form of emails but also in various other sources, for example, spam comments on a web page and so on and same techniques we apply to email spam detection could also be applied to detecting spam in various documents.

H. Classification Algorithm

Classification algorithms learn by example using data which is organized into classes through some automated process or manually. Through a training process, the classification algorithm determines which features (or properties) indicate that an item belongs to a given class of objects. After the training process, classification algorithms can classify data which was unlabeled earlier. There are many classification algorithms and are differentiated by the output they produce for example Binary algorithms produce a yes or no depending upon an object belonging to a class or not, other algorithms support multiple outcomes.

Spam filtering is an example of binary classifiers. Binary classifier identifies an email as spam depending upon the features of that email. The classification process consists of multiple phases as shown in the figure below

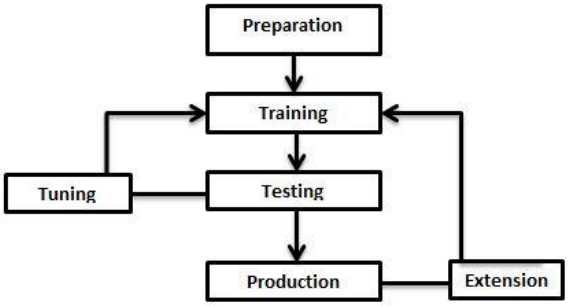

Figure 2: Classification process

Various steps involved in the classification process (irrespective of algorithms) are described as follows:

1. Preparation: This phase involves preparing data which is used for the training process. In this phase set of labels are chosen which our classifier will be trained to identify, the way in which the features (or properties) used for training are identified. Next, data is transformed into the format used by the training algorithm.
2. Training: The data from the preparation phase acts as an input to training phase. In this phase training algorithm processes each labeled example and identifies how features of each relate to its label. Each feature is associated with the label assigned to the document and the training algorithm models the relationship between features and class labels.
3. Testing: In this phase, the classification algorithm is evaluated using test data. The evaluation process compares the class each example belongs to with the class assigned by the classifier. Accuracy of the training algorithm is determined by using a number of correct and incorrect class assignments. This output is then used as feedback in adjusting the parameters and techniques used in the preparation and training phases.
4. Production: As the name implies in this phase our classifier is put into production. A classifier may be trained multiple times, and it's very common to repeat the training and testing phase for producing better results.

1)BAG-OF-WORDS: One of the most common and effective ways of representing text for machine learning is Bag-of-words (BoW). This representation is used for document categorization, for example, classifying various blog articles in different categories such as education, entertainment, sports etc. In bag-of-words representation, most of the input structure of the text like chapters, sentences, formatting etc., is discarded and only the number of occurrences of each word appearing in the text is taken into consideration. In this representation linguistic structure of the text is not considered and BoW concerns itself only with the presence of known words in the document, ordering of words is also ignored.

2) Naïve BAYES CLASSIFIER: The Naive Bayes algorithm is a probabilistic classification algorithm; it is used for text classification involving training data sets, and frequently used for multi-class or binary classification. This algorithm is probabilistic because it decides assigning a class to an input document using probabilities derived from training data. The training process analyzes the relationships between words in the training documents and categories and the relationships between categories and the entire training set. The available facts are collected using calculations based on Bayes' Theorem to produce the probability that a collection of words (a document) belongs to a certain class. This algorithm is termed as naïve because it assumes that words appearing in a given class are independent of each other. Through our human experience, we know that words don't occur independently in a document belonging to a specific subject for example words like "monkey" are more likely to occur in documents containing the word "jungle" rather than the word "water". Probabilities produced by the naive Bayes algorithm aren't true probabilities, but they are useful as relative measures.

3) Bayes Theorem: Naive Bayes classifier uses the Bayes Theorem. Bayes theorem works on conditional probability. Conditional probability is the probability that something will happen, given that something else has already occurred. Using the conditional probability, we can calculate the probability of an event using its prior knowledge. Bayes' theorem is stated mathematically as the following equation:

$$P(A|B) = \frac{P(A)*P(B|A)}{P(B)}$$

Where:

- A & B are events
- P(A) and P(B) are the probabilities of A and B without regard for each other
- P(A|B) is the conditional probability, the probability of A given that B is true
- P(B|A) is the probability of B given that A is true

III. PROPOSED SYSTEM

Our proposed system is divided into two parts:

A. Part 1: First part demonstrates various NLP techniques using Core Java, Lingpipe[5] and OpenNLP[6]. Lingpipe and OpenNLP are open source libraries for performing various natural language processing tasks, specialized libraries like OpenNLP and LingPipe are required because they provide various features named entity recognition, parts-of-speech tagger etc not found in programming languages like Java. For text extraction from Adobe PDF files, PDFBOX[7][8] an open source java library is used. For HTML parsing JSOUP[9][10] is used, it's a Java-based library and provides a very handy API for extracting and manipulating HTML data, using the best of DOM, CSS, and JQuery like methods.

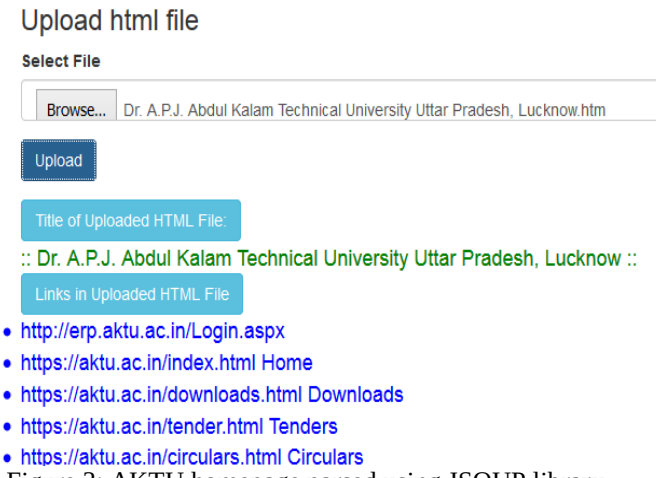

Figure 3: AKTU homepage parsed using JSOUP library

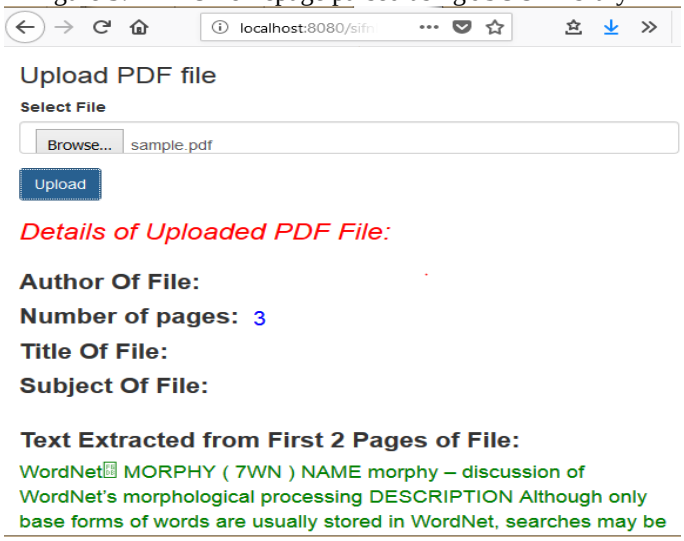

Figure 2: Result after parsing sample PDF file using PDFBox

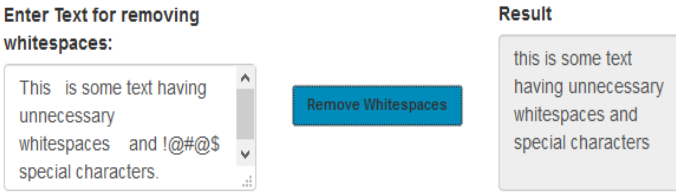

Figure 3: Removing whitespaces and special characters from text

Figure 4: Stop word removal using Core Java vs. LingPipe

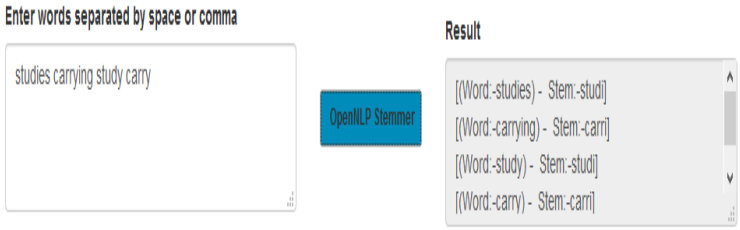

Figure 5: Word-Stem result

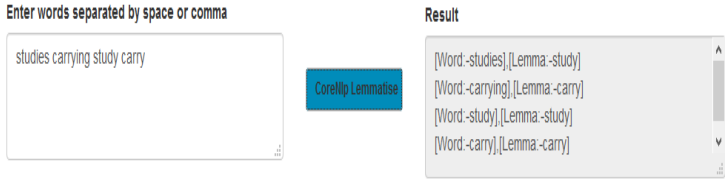

Figure 6: Word-Lemmatize Result

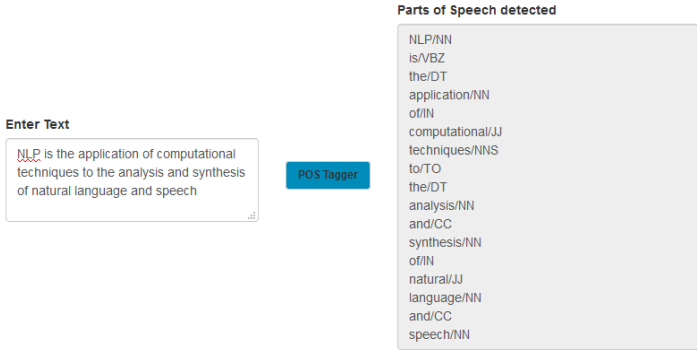

Figure 7: POS Tagging Demo

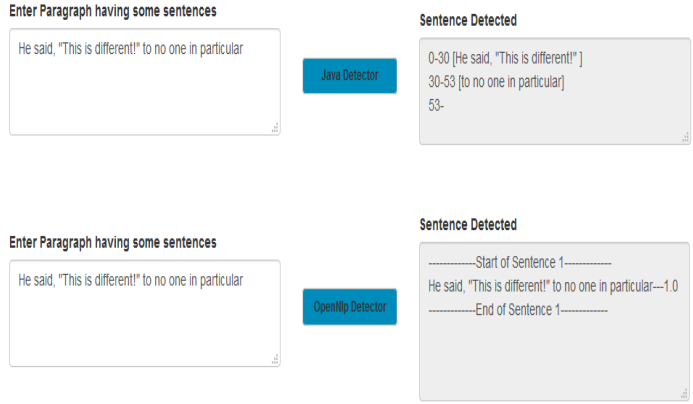

Figure 8: Java/OpenNLP results for sentence detection

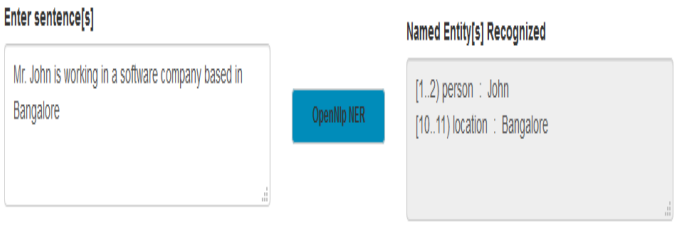

Figure 9: Named Entity Recognition

B. Part 2: Second part of the proposed system demonstrates implementation of E-Mail spam filtering using MALLET[11] (MAchine Learning for LanguagE Toolkit). MALLET is a tool written in Java for machine learning applications like natural language processing, clustering, document classification, topic modeling and information extraction to texts. It is a product of the University of Massachusetts Amherst, and it's written by Andrew McCallum and a team of collaborators. The email spam dataset used in the proposed system is collected by Androutsopoulos [12] and it is one of the first e-mail spam datasets for benchmarking spam-filtering algorithms. They studied how the naive Bayes classifier can be used to detect spam. The dataset was reorganized by Andrew Ng in OpenClassroom's machine learning class[13]. This dataset is available in zip format. Compressed file contains four folders, the nonspam-train, and spam-train folders contain the pre-processed e-mails which the proposed system uses for training, each folder has 350 e-mails.

1. MALLET initialization: Once compressed emails are extracted on a specific location, an iterator is initialized that runs through email spam dataset; data is transformed (by applying a sequence of MALLET-pipelines for cleaning the data and converting the data into the format which can be used by MALLET). Mallet supports this process through a pipeline. Preprocessing the data helps in improving result output, for example, applying stemming and removing stop words reduce the feature space dimension and possibly improve the prediction accuracy of the classifier by alleviating the data sparseness.

Few of MALLET pipeline classes which are used in the code are described below

- Input2CharSequence: Pipe for reading from different kinds of text sources (File or URI) into CharSequence.
- CharSequence2TokenSequence: Pipe for tokenizing a character sequence.
- TokenSequenceLowercase: This pipe converts the text of each token to lower case.
- TokenSequenceRemoveStopwords: Remove tokens based on stop word list.
- TokenSequence2FeatureSequence: Converts the token sequence from the data field of each instance to a feature sequence.

2. Training model and testing: Mallet implements a set of classifiers in the cc.mallet.classify package, including decision trees, naive Bayes, AdaBoost, bagging, boosting, and many others. For our purpose a basic classifier, that is, a naive Bayes classifier is used. A classifier is initialized by the ClassifierTrainer class, which returns a classifier when its train(Instances) method is invoked. Next performance of the classifier is evaluated on a separate dataset.

3. Model Performance

1) To evaluate the classifier on a separate dataset, emails located in test folder are imported.
2) To evaluate classifier performance, cc.mallet.classify.Trial class is used, which is initialized with a classifier and set of test instances.
   a. Trial trial = new Trial(classifier, testInstances);

4. The evaluation is performed immediately at initialization. We can then simply take out the measures that we care about. In our proposed system, we'd like to check the precision and recall on classifying spam e-mail messages, or F-measure, which returns a harmonic mean of both values, On running the application result we get are displayed in the following figure:

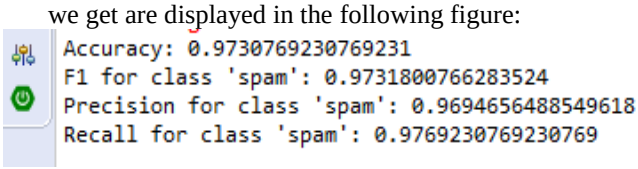

Figure 12: Result

As is displayed in the result, MALLET detects about 97% of emails as spam which is very close to the expected result.

IV. CONCLUSION AND FUTURE WORK

We discussed the importance of unstructured data and immense value it gives in providing high-quality information when proper analysis techniques are applied. We looked at few open source tools to extract text from proprietary formats and how to clean data collected from different sources before we could start any processing on it. We then looked at various natural processing techniques to add some structure to our data to make it intelligible for various text mining tasks. We also discussed differences between text mining and natural language processing. Finally applying all these techniques we used naïve Bayes classifier to detect email spam.

With the increased use of the Internet, both natural language processing and text mining have become more and more important. In NLP we are mostly concern with the semantic and grammatical structure of text but in text mining semantics are not considered, so a scientific assessment is required for finding which linguistic concepts and NLP techniques are beneficial for which text mining applications. This would involve a clear classification of the various linguistic concepts that might be of use (e.g. part-of-speech, grammatical role) and the various technologies for getting hold of these concepts, as well as a classification of text mining.

There are also many open areas of research in text mining and natural language processing, for example we have implemented a text based spam detector, but nowadays spam mail have images rather than text as main body. One other important open area of research in field of natural language

processing is designing an effective questioning and answering systems, for example when user enters keyword in Google the user is presented with a list of web pages/documents, using natural language processing we can design a tool which may answer user's query in form of simple text based answer, such type of few systems exists but a lot more research needs to be done in this field.

REFERENCES

[1] John Gantz and David Reinsel. (2011, June) EMC Corporation. [Online]. http://www.emc.com/collateral/analyst-reports/idc-extracting-value-from-chaos-ar.pdf

[2] E.D. Liddy, "Natural Language Processing," Encyclopedia of Library and Information Science, vol. II, 2001.

[3] David A. Hull. (1995, June) Stemming Algorithms - A Case Study for Detailed Evaluation. [Online]. www.inf.ed.ac.uk/teaching/courses/tts/papers/hull96stemming.pdf

[4] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz, "Building a Large Annotated Corpus of English: The Penn Treebank," 1993.

[5] Jeffrey Reynar and Adwait Ratnaparkhi, "A Maximum Entropy Approach to Identifying Sentence Boundaries," Department of Computer and Information Science,University of Pennsylvania,.

[6] http://alias-i.com/.

[7] Apache OpenNLP. [Online]. http://opennlp.apache.org/

[8] Apache PDFBox. [Online]. https://pdfbox.apache.org/

[9] Duy Duc An Bui, Guilherme Del Fiol, and Siddhartha Jonnalagadda, "PDF text classification to leverage information extraction from publication reports," Journal of Biomedical Informatics, no. 61, pp. 141-148, April 2016.

[10] JSOUP. [Online]. https://jsoup.org/

[11] Ayar Pranav and Sandip Chauhan, "Efficient Focused Web Crawling Approach for Search Engine," IJCSMC, vol. 4, no. 5, pp. 545-551, May 2015.

[12] MAchine Learning for LanguagE Toolkit. [Online]. http://mallet.cs.umass.edu/

[13] Androutsopoulos Ion, John Koutsias, Konstantinos V. Chandrinos, George Paliouras, and Constantine D. Spyropoulos, "An Evaluation of Naive Bayesian Anti-Spam Filtering," in 11th European Conference on Machine Learning, Barcelona, 2000, pp. 9-17.

[14] openclassroom.stanford.edu. [Online]. http://openclassroom.stanford.edu/MainFolder/DocumentPage.php?course=MachineLearning&doc=exercises/ex6/ex6.html

