International Journal of Computer Sciences and Engineering Open Access
Survey Paper Volume-6, Special Issue-3, April 2018 E-ISSN: 2347-2693

A Survey on Text Pre-processing Techniques and Tools

Ravi Lourdusamy[1], Stanislaus Abraham[2*]

[1]Dept. of Computer Science, Sacred Heart College, Thiruvalluvar University, Tirupattur, 635 601, Tamil Nadu, INDIA
[2]Dept. of Computer Science, Sacred Heart College, Thiruvalluvar University, Tirupattur, 635 601, Tamil Nadu, INDIA

[*] Corresponding author: astanislaus@gmail.com, Tel.: +91-9489914557

***Abstract*—** We live in an era of digital data explosion over Internet. Data warehouses deal with numerical databases than textual sources. Nearly eighty percent of digital data is either in semi or un-structured textual form. Several knowledge mining techniques developed over the past decade and those that are being developed now continue to draw attention to transform such textual data into desirable information and useful knowledge. This knowledge and information is used to benefit many fields of applications such as: social network, business management, customer care management system, market analysis, search engines, fraud detection, just to name a few. Text Mining (TM) is what is needed if desired information is to be obtained from such voluminous data. TM is multi-disciplinary in nature. Several TM techniques are deployed in the process of extracting knowledge from textual sources. Input text for such techniques needs to be pre-processed and cleaned. This survey briefly presents pre-processing tools for TM in general and Natural Language Processing (NLP) in particular. Also presents the broad categories of TM techniques used. The focus of this paper is to explore and analyze several features of text preprocessing techniques and tools that would interest researchers in the area of TM.

***Keywords*—**Text Mining, Pre-processing techniques, Pre-processing Tools, Natural Language Processing

I. INTRODUCTION

Text is a leading medium for exchange of information, be it with humans, from time immemorial and of late, with machines too. The aim of text-mining applications is to trace, retrieve and operate on data of relevant information efficiently from the volume of text which continues to increase exponentially and expeditiously.

TM as knowledge discovery process, first made known by Fledman [1] is an extraction of previously unknown facts by mining information from textual sources. It uses techniques and procedures from various specialized areas such as statistics, machine learning (ML), data mining (DM), natural language processing (NLP), information retrieval (IR), and information extraction (IE). Text Mining typically consists of three important phases namely information retrieval, information extraction and data mining. IR is an approach fundamentally corresponding to congregation of information after applying a process of filtering on the relevant documents. IE techniques prefer explicit facts about pre-specified types of entities and relationships. DM is a technique that discovers unassuming associations between known

facts. An overview of TM process is shown below in figure 1.

An enormous amount of data is in textual form. To perform text mining, the textual data is processed in which different refinement techniques are applied to arrive at a refined data. Then this refined data is converted into a form that will be more suitable to extract knowledge from the given text. This practice is titled as "pre-processing" of text [2].

Research community working on different pre-processing procedures uses Tokenization, Stop Word Filtering, Parts-Of-Speech Tagging, Lemmatization, Stemming, Document Indexing, Grammatical Parsing & Chunking as pre-processing techniques.

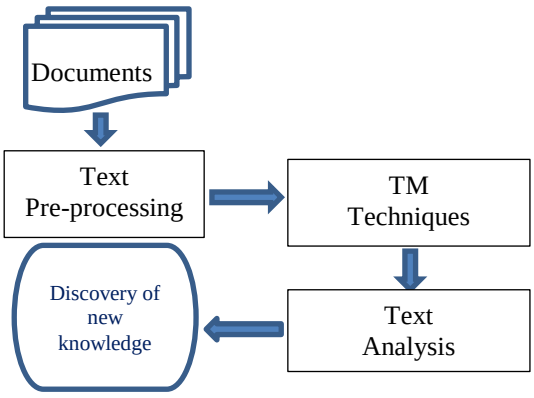

Fig. 1 An Overview of TM Process

This survey briefly presents text pre-processing tools with TM techniques. The objective of this study is to explore and analyze features and tools that would interest researchers in the area of TM.

The presentation of the study is summed up as following: In section 2 various TM pre-processing techniques are presented. In section 3 survey on related works is brought out. Section 4 deals with text pre-processing tools taken in our study of research. Feature analysis of text pre-processing tools is furnished in section 5. Finally, section 6 concludes the survey on text pre-processing techniques and tools.

II. TEXT PRE-PROCESSING TECHNIQUES

Pre-processing techniques are vital in TM and its applications. It is also called cleaning the text, which is performed to identify and eliminate glitches in the given text. Various pre-processing techniques are available to prepare the text to be ready for further analysis and mining process to achieve significant performance. Most of the core functionalities of pre-processing techniques deal with NLP. The following are some major pre-processing techniques: Tokenization, Stop Word Filtering, Parts-of-Speech (POS) Tagging, Word Sense Disambiguation (WSD), Grammatical Parsing and Chunking, Lemmatization, Stemming, Text Summarization, and Term Frequency and Inverse Document Frequency (TF-IDF).

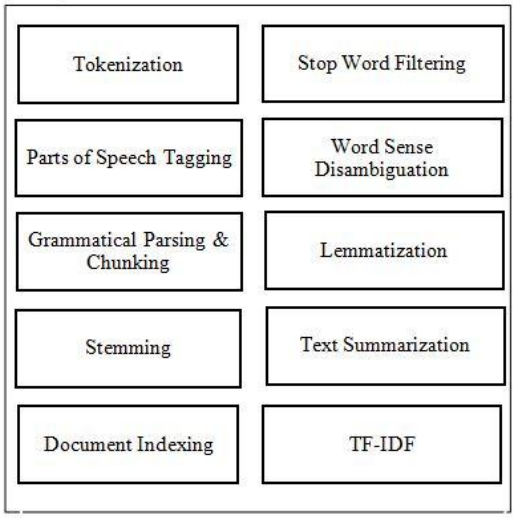

Fig 2. Text Pre-processing Techniques

A. Tokenization

It is a method of segmenting a group of text into meaningful elements, or tokens, such as: words, phrases, and symbols. The list of collection of tokens is used as input text for further processing. Challenges in the process of tokenization are governed by the type of the natural language. On the one hand, most of the natural languages use white spaces as delimiter between words and on the other a few other natural languages do not have clear boundaries between words.

B. Stop Word Filtering

Stop words are frequently used common words like 'and', 'are', and 'this'. These stop words do not contribute to the knowledge source and they can be removed from the textual data. Challenges in the process of stop word filtering is firstly, difficulty in constructing a list of stop words because of its inconsistency between different textual sources and secondly their high frequency of occurrences pose difficulty in processing the textual data.

C. Parts-of-Speech Tagging

POS enhances the 'word' and its 'context' with huge volume of information about itself and its neighbors. The usage of a word in a sentence such as: noun, pronoun, verb, adverb, adjective, article, preposition, and conjunction, helps to infer possible knowledge about neighboring words and syntactic structure weaving around the word. Thus POS tagging becomes an inseparable component in syntactic parsing.

POS tagging is the process of assigning a POS marker to each word in an input text. The input to a tagging algorithm is a sequence of words and a tag set, and the output is a sequence of tags, and a single best tag for each word.

Tagging is a disambiguation task; since words having more than one POS are used in a sentence, the aim is to tag the word with right parts-of-speech. For example, the word book can be a verb (book that flight) or a noun (as in hand me that book) and 'that' can be a determiner (Does that flight serve dinner) or a complementizer resolution (I thought that your flight was earlier). The problem of POS-tagging is to resolve these ambiguities, choosing the proper tag for the context.

D. Word Sense Disambiguation

"Words may refer to several contexts, for example the word horse may refer to an animal, to a unit of horsepower, to heroin, to an obstruction in a vein (mining), and to a frame or structure on which something is mounted" [3]. Disambiguation is a process where each term in the corpus is assigned to a specific context. WSD is the task of finding meaning intended by the word in a context. It has been a long-standing research objective for NLP. WSD is contextual, i.e., group of words appearing next to each other in the same context apparently have related meanings [4].

Natural language has an inherent complexity of expressing an idea or concept in myriad ways. Term becomes ambiguous when a single word is used to refer to manifold concepts. Words have multiple synonyms. Word meaning gets changed by its context. Sub-languages provide context confined to specialized domains, [5] that normally brings down the level of ambiguity.

E. Grammatical Parsing & Chunking

Grammatical parsing analyses each word in a sentence to determine structure from its constituent parts. It does so, using two components viz., parser and grammar. Parser is a computer program having a procedural component. It does not change its procedure be it any language being used. The grammar can change depending on the language being used. So, a system can parse any language by changing its grammar. NLP parser develops the grammatical structure of the sentence, for example, a group of words which is a series of words (phrases) and which word is the subject or objects of a verb and generates the grammar tree of it. While performing parsing, several specific type of grammar rules are used [6]: a) Context-Free Grammar, and b) Lexicalized Context-Free Grammar [7].

In formal language and automata theory, Context-Free Grammar is defined as a set of production rules that generate all possible patterns of strings in a given language. It consists of terminal symbols, nonterminal symbols, productions and a start symbol as its components. In Context-Free Grammar, all rules are one-to-one, one-to-many, or one-to-none. Lexicalized Context-Free Grammar is an outcome of amalgamation of parsing efficiency of Context-Free Grammar and a restricted, elegance form of lexical sensitivity of lexicalized tree adjoining grammar. It consists of a set of initial trees and a set of auxiliary trees.

Chunking is similar to parsing. It is used to build hierarchical structure over text. It is not exhaustive. It ignores some items in the surface string. Sometimes it is called partial parsing.

F. Lemmatization

The objective of stemming and lemmatization is to arrive at a common stem by minimizing inflectional forms of words and derived related forms of a word. Stemming refers to a basic heuristic process that truncates the ends of words. It includes the removal of derivational affixes.

Lemmatization refers to the use of a vocabulary and morphological analysis of words, aiming to remove inflectional endings and to return the base form of a word, which is referred as lemma. If confronted with the token *saw*, stemming might return just *s*, whereas lemmatization would attempt to return either *see* or *saw* depending on whether to use of the token was as a verb or a noun. Stemming most commonly fail with the derivationally related words. Lemmatization fails in the various inflection forms of a lemma.

G. Stemming

Stemming is the process of identifying a root word that forms into a different representation. For example, the words: “transportation”, “transported”, “transporting” could all be identified by the root word or stem word “transport”. Challenges of stemming depend on the process of controlling the errors.

Over stemming (false positive) and under stemming (false negative) are common errors in stemming. Over-stemming is when two words with different stems are stemmed to the same root. Under-stemming is when two words that should be stemmed to the same root are actually stemmed to different words. For example, the widely used Porter stemmer stems "*universal*", "*university*", and "*universe*" to "*univers*". “This is a case of overstemming: though these three words are etymologically related, their modern meanings are in widely different domains, so treating them as synonyms in a search engine will likely reduce the relevance of the search results. An example of understemming in the Porter stemmer is "*alumnus*" → "alumnu", "*alumni*" → "*alumni*", "*alumna*"/"*alumnae*" → "*alumna*". This English word keeps Latin morphology, and so these near-synonyms are not conflated.

H. Text Summarization

TM applications summarize all sorts of text documents and a collection of documents on a specific topic to arrive at a concise overview [8, 9].Summarization follows two techniques: viz., extractive and abstractive. Extractive summarization contains information units extracted from original text, and on the other hand abstractive summarization brings out a summary of “synthesized” information that may or may not be found in original document [10, 11].

I. Document Indexing

“A set of terms to be used in a document are used for indexing the document. It is based on choosing suitable set of keywords based on the whole corpus of documents, and assigning weights to those keywords for each particular document, thus transforming each document into a vector of keyword weights. The weight is related to the frequency of occurrence of the term in the document and the number of documents that use that term” [12].Various approaches are used for selecting index terms. One such approach is the identification of noun groups. “Combining two or three nouns in a single component (e.g., Computer Science, European Union, United Arab Emirates), makes sense to cluster nouns which appear nearby in the text into a single indexing component (or concept). A noun group is a set of nouns whose syntactic distance in the text does not exceed a predefined threshold” [13].

J. Term Frequency and Inverse Document Frequency (TF-IDF)

Tf-idf weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the

number of times a word appears in the document but is offset by the frequency of the word in the corpus. The tf-idf weight is composed by two terms: the first computes the normalized Term Frequency (TF), the number of times a word appears in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

Where $n_{i,j}$ is the number of occurrences of the considered term (ti) in document d_j , and the denominator is the sum of number of occurrences of all terms in document dj , that is, the size of the document |dj|.

IDF(t) = log_e(Total number of documents / Number of documents with term t in it).

$$idf_i = log \frac{|D|}{|\{j: t_i \in d_j\}|}$$

with |D| : cardinality of D, or the total number of documents in the corpus $|\{ j : ti d_j\} \in$ number of documents where the term t_i appears (viz., the document frequency) (that is $n_{i,j} \neq 0$). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to use 1 + $|\{j : ti \quad d_j\}|$. $\in$

III. RELATED WORKS

Most of the research work elicit the text pre-processing techniques and its relevance in TM[12-20].A few research works have made study on the effect of pre-processing techniques on text categorization, using various machine learning algorithms [12]. Stemming algorithms and its classification are studied implementing improved Porter's algorithm and comparing with Krovetz's algorithm [14, 19].

Vijayarani et al., (2015) have presented an exhaustive study on pre-processing techniques. Further, the research work brings out the classification of the stemming algorithms into three groups; viz., truncating, statistical and mixed methods [15]. Vijayarani et al., (2016) have analysed the performance of tokenization tools two measures viz., limitations and output are used for the comparative study. The authors conclude NLP.net tokenizer is the best among the tools studied. The authors have felt the need to develop tokenizers for all the natural languages [16].

Swapnil Vaidya et al., (2017) discusses about preprocessing techniques for NLP in social media. Tweet normalization, hashtag segmentation, named entity recognition are the preprocessing techniques used for social media. The authors have proposed to do Sentiment Analysis on the results of preprocessing techniques [17]. Nikita P.K. et al., (2015) elicits methods for mining text data with the use of side information such as web logs, links, and meta-data. The authors conclude that the domain specific applications are more suitable for text mining [13, 16, 18].

Mohd Zakree Ahmad Nazri et al., (2008), deliberated upon overall process of learning taxonomy from Malay texts using unsupervised conceptual clustering approach and probed into the then prevailing Malay NLP tools as potential pre-processing tools for the proposed ontology learning approach. Following tools were used for their exploratory case study: maximum-entropy parser based on open NLP package, a word sense tagger and a parser based on pola grammar. The outcome showed lower recall and precision [20].

IV. REVIEW OF TEXT PREPROCESSING TOOLS

A. TextBlob

TextBlob is a tool for processing textual data. It is implemented in Python library. It provides a simple API for plugging into several common NLP tasks. The various features supported by TextBlob are Add new models or languages through extensions, Classification, Language translation and detection, Noun phrase extraction, Parsing, n-grams, Part-of-speech tagging, Sentiment analysis, Spelling correction, Tokenization, Word and phrase frequencies, Word inflection and lemmatization, and WordNet integration.

B. Pattern

Pattern is Python based web mining module. It has tools for visualization techniques, NLP, ML, DM, and network analysis. Text is mined from the web and searched by syntax and semantics sentiment analysis is performed on matching phrases.

This tool has five modules, viz., web module, en module, search module, vector module and graph module. The web module is a tool kit that contains data mining (APIs, HTML DOM parser and a web crawler), natural language processing (POS taggers, preprocessing, n-gram search, sentiment analysis, WordNet), machine learning (vector space model, clustering, SVM), network analysis and canvas visualization. The en module is a tool kit for English and has POS tagger, word inflection and a WordNet API. The search module consists of search algorithms to retrieve n-grams. The vector module

consists of algorithms for clustering, classification and semantic analysis. The graph module: It provides a visual representation of possible nodes (terms and concepts).

C. Word Tokenization with Python NLTK

Natural Language Tool Kit provides packages which can be imported into Python programme. It provides interfaces to several lexical resources such as WordNet along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning.

NLTK provides a number of tokenizers in the module. The text is first tokenized into sentences using the PunktSentenceTokenizer. Then each sentence is tokenized into words using four different words tokenizers. TreebankWordTokenizer uses regular expressions to tokenize text as in Treebank. WordPunctTokenizer divides a string into substrings by splitting on the specified string, which is defined in subclasses. PunctWordTokenizer divides a text into a list of sentences by using unsupervised algorithms. WhitespaceTokenizer divides text at whitespace.

D. MILA Tokenizer

MILA was developed in 2003 by the Israel Ministry for Science and Technology. Its mission is to create the infrastructure necessary for computational processing of the Hebrew language and make it available to the research community as well as to commercial enterprises. MILA has acquired a number of Hebrew corpora from various domains. MILA morphological analytical tool takes undotted Hebrew input text and returns tokens, morphological analysis of the tokens, reflecting POS, transliteration, gender, number, definiteness, and possessive suffix.

The MILA Morphological Disambiguation Tool takes as input morphological analyzed text and uses Hidden Markov Model (HMM) to assign scores for each analysis considering contextual information from the rest of the sentence.

E. StanfordCoreNLP for .NET

StandfordCoreNLP implemented using .NET, is an integrated framework which provides a set of natural language tools for English language. It integrates all Stanford NLP tools including the POS tagger and Named Entity Recognizer, the parser, the co-reference resolution system and the sentiment analysis tools.

F. sharpnlp

sharpnlp is a collection of NLP tools written in C#. It provides sentence splitter, tokenizer, POS tagger, chunker, parser, name finder, co-reference tool and an interface to WordNet lexical database.

G. Lexalytics

Lexalytics is a text analytics tool. It breaks sentences and phrases to evaluate semantics, syntax, and context. Supervised, semi-supervised machine learning, and NLP techniques are deployed to perform sentiment analysis to extract named entities, themes, categories and intentions and thus to create summaries. The complex relationship within the text is revealed as connections between entities, themes, sentiment of individual entities and categories to offer rich context insights.

H. SAS Text Miner

This TM software lets one analyze easily text data from the web, comment fields, books and other text sources. Visual interrogation of results, native support for multi-lingual, term profiling are the important features supported by the software.

I. Aika

Aika is a Java library that automatically extracts and annotates semantic information into text. It allows to model linguistic concepts like words, entities, categories, and grammatical word types as neurons in a neural network. By choosing appropriate synapse weights, these neurons can take on different functions within the network. Aika generates multiple mutually exclusive interpretations of a word, phrase, or sentence and select the most likely interpretation. It has two layers, viz., neural layer; logic layer. The neural layer consists of a Boolean representation of all the neurons. The logic layer uses lattice to store the individual logic nodes.

J. Open NLP

The Apache OpenNLP library is a machine learning based toolkit for the processing of natural language text. It supports the most common NLP tasks, such as tokenization, sentence segmentation, POS tagging, named entity extraction, chunking, parsing, and co-reference resolution. These tasks are usually required to build more advanced text processing services. The Apache OpenNLP Library contains several components such as sentence detector, tokenizer, name finder, document categorizer, part-of-speech tagger, chunker, parser, and co-reference resolution. It also consists of an API which is accessible by all the components and a command line interface for conducting experiments.

K. Rosette

Its a multilingual text Analytics tool. It fetches the power of Artificial Intelligence to text analysis. Its applications are in business intelligence, e-discovery, social media, financial compliance, and enterprises. Its strength lies in

speed and accuracy in analyzing massive volumes of data to enable multilingual analysis or to build other specialized AI applications.

Rosette's value is in enabling applications to manipulate text like numbers. The following are its capabilities: Categorization, Entity Extraction & Linking, Language Identification, Morphological Analysis, Name Matching, Name Translation, Relationship Extraction, Sentence Tagging, Sentiment Analysis, Tokenization, and Topic Extraction. This tool is integrated with curl, implemented in Python, PHP, JAVA, C#, nodejs, and Ruby.

L. Buzzlogix

It has a distinctive set of features. Social media specialists using these features simplify campaigns related to social media, automate tasks, measure results, extract valuable insights, and make useful recommendations throughout the process.

It brings advanced automation to its All-In-One Social Media Monitoring, Analytics and Engagement solution. The following are its special features: Classification, Entity Extraction, Gender Detection, Keyword Extraction, Sentiment Analysis, Subjectivity Analysis, and Twitter Sentiment Analysis.

M. KBSPortal

It's a NLP Library coded in Ruby. This portal is used for tagging and categorizing automatically content submitted by user on web site, to identify named entities in text, using ngram analysis and Latent Semantic Indexing to generate summaries of text automatically, to link related documents by people, products and companies discussed in the documents, to cluster documents with similar nature and to rate user content by sentiment.

N. meaningcloud

This tool extracts semantic meaning from unstructured content like articles, documents, social conversation, etc. The special features are as follow: Corporate Reputation, Deep Categorization, Document Structure Analysis, Language Identification, Lemmatization, POS and Parsing, Sentiment Analysis, Summarization, Text Classification, Text Clustering, and Topics Extraction.

O. megaputerTextAnalyst

It is a unique software tool for semantic analysis, navigation, and search of unstructured texts. This tool helps to meet the ever increasing challenges of skimming and scanning through large volumes of textual data to arrive at a decision. It is an intelligent electronic tool to extract textual meaning in a concise form, to inspect accurate summaries, to traverse through large text bases and to accomplish NLP IR. Many such tasks are efficiently handled by TextAnalyst. A synergy of unique linguistic and neural network technologies implemented in TextAnalyst ensures high speed and accuracy in the analysis of unstructured texts.

P. monkeylearn

This tool helps to turn tweets, emails, documents, webpages and more into actionable data. Automate business processes and save hours of manual data processing. Also to analyze text data to tag customized categories for obtaining actionable insights, to automate workflows ranging from marketing, sales and customer service. It also automates business processes. A special feature is implementation of NLP for developers in monkeylearn. Monkeylearn APIs are available to interface with languages like: -python, ruby, node, php, and java.

Q. PolyVista Text Analytics

It simplifies data obtained from various sources like, social media, call center data, chats, returns feedback, survey responses, etc. to unearth similarity, relatedness, and common patterns existing in the data. Features such as sentiment analysis and classification of data are special to this tool.

R. Skyttle

It is a Software as a Service tool that offers services to extract patterns from text in text analytics and stock them in a structured format for an exhaustive data analysis in future. It integrates text analytics into applications with complex NLP functionalities into software, and gain fast and valuable insights into large text collections.

Its features are Customizable to specific applications, Domain customization, Easily scalable, Keyword extraction, Named Entity Recognition, Phrase-level Sentiment Analysis, Sentiment analysis, Terminology/Keyword/Concept Extraction, XML annotation.

V. FEATURE ANALYSIS OF TEXT PRE-PROCESSING TOOLS

Various pre-processing techniques and tools available for TM were taken for an in-depth study. Feature analysis with respect to the tools studied was carried out highlighting the various features of the tools. The URL of the tool along with NLP and TM techniques and the language of implementation were summarized in the form of a table 1.

This article presents various text pre-processing techniques and the tools implementing these techniques that might be useful to the researchers in order to improve the performance of the various preprocessing techniques and to enhance the existing tools or to innovate new ones. Selection and use of right pre-processing techniques and

tools according to the domain might help to make the text pre-processing easy and efficient.

A binary matrix is constructed using the table 1 and a Rank Order Clustering algorithm is applied for deducing the following findings:

i) Classification, Language translation and detection, Noun phrase extraction, Parsing& Chunking, Parts-of-speech tagging, Sentiment analysis and Tokenization are commonly available in most of the tools surveyed.
ii) TextBlob, meaningcloud, KBSPortal, Lexalytics, Skyttle, StanfordCoreNLP for .NET, SAS Text Miner, Aika, Open NLP, sharpnlp, MILA Tokenizer, Rosette, and Buzzlogix are the popular tools supporting the commonly available techniques. The tools are ordered according to the number of features supported by each tool. For example TextBlob is a tool which supports nine features whereas Buzzlogix supports five only. The comparison of features supported by the tools is exhibited using a graph as shown in Figure 1.

iii) Most of the preprocessing tools are implemented in Python.

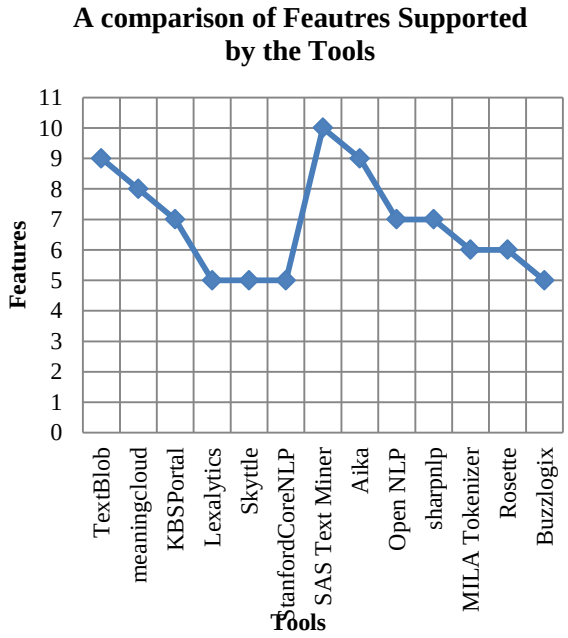

Figure 1: A Comparison of features supported by tools

S.No.	Name of the Tool URL	NLP and Text Pre-processing Techniques/Features														
		Noun phrase extraction	Parts-of-speech tagging	Sentiment analysis	Classification	Language translation and detection	Tokenization	Parsing& Chunking	n-grams	Add new models or languages through extension	Lemmatization\Stemming	Semantic Reasoning/WSD	Graph analysis & visualization.	Sentence Segmentation	co- reference resolution system	Language Used
1	**TextBlob https://pypi.python.org/pypi/textblob**	X	X	X	X	X	X	X	X	X						Python
2	**Pattern https://www.clips.uantwerpen.be/pattern**		X	X					X	X			X			Python
3	**Word Tokenization with Python NLTK http://text-processing.com/demo/tokenize/**		X				X	X			X	X				Python
4	**MILA Tokenizer http://www.mila.cs.technion.ac.il/tools_token.html**	X	X				X	X				X	X			JSP
5	**Stanford CoreNLP for .NET www.sergey-tihon.github.io/Stanford.NLP.NET/StanfordCoreNLP.html**	X	X	X				X							X	.NET
6	**sharpnlp www.sharpnlp.codeplex.com**	X	X				X	X		X				X	X	C#
7	**Lexalytics https://www.lexalytics.com/**	X	X	X	X			X								Ruby, .NET, JS, python, JAVA,C++ and PHP

S.No.	Name of the Tool URL	NLP and Text Pre-processing Techniques/Features														
		Noun phrase extraction	Part-of-speech tagging	Sentiment analysis	Classification	Language translation and detection	Tokenization	Parsing& Chunking	n-grams	Add new models or languages through extension	Lemmatization\Stemming	Semantic Reasoning	Graph analysis & visualization.	Sentence Segmentation	co- reference resolution system	Language Used
8	**SAS® TEXT MINER https://www.sas.com/en_us/software/text-miner.html**	X	X		X	X	X	X		X	X	X	X			C Language
9	**Aika http://www.aika-software.org/**	X	X		X	X	X	X				X		X	X	Java
10	**OpenNLP https://opennlp.apache.org/**	X	X			X	X	X						X	X	JSP
11	**Basic Technology Rosette https://www.rosette.com/**	X		X	X	X	X							X		Integrated with curl, Python, PHP, JAVA, C#, nodejs, Ruby
12	**Buzzlogix text analysis api https://buzzlogix.com/press**	X		X	X	X	X									Ruby, .NET, JS, python, IOS, GO Lang and PHP
13	**Kbsportal http://kbsportal.com/**	X	X	X	X	X	X		X							Ruby
14	**Meaningcloud https://www.meaningcloud.com/**	X	X	X	X	X	X	X			X					PHP, Java, GO Lang, Python
15	**Megaputer Text Analyst https://www.megaputer.com/site/textanalyst.php**				X							X				C++, Java, JavaScript
16	**Monkeylearn https://monkeylearn.com/**			X	X											monkeylearn-python, -ruby, -node, -php, -java
17	**Polyvista http://www.polyvista.com/Home/**			X	X											
18	**Skyttle APIhttp://www.skyttle.com/**	X	X	X			X	X								Java

V. CONCLUSION

The main challenging issue in TM arises from the complexity of a natural language itself. This survey has briefly presented pre-processing tools for TM with broad categories of TM techniques used. The focus of this paper was to explore and initiate a discussion on feature analysis of text preprocessing techniques and tools that would interest researchers in the area of TM.

Integrating domain knowledge base with TM would increase the efficiency of the IR & IE techniques. There is a dire need for a powerful tool integrating all the preprocessing techniques as one product. Multilingual preprocessing tools are very few in the field and hence there is more avenue to do further research in this area.

REFERENCES

[1] Feldman Ronen & Dagan Ido, "Knowledge Discovery in Textual Databases", KDD, Vol. 95. pp. 112–117, 1995.

[2] Saira, Gillani Andleeb, "From text mining to knowledge mining: An integrated framework of concept extraction and categorization for domain ontology", PhD Dissertation, Budapesti Corvinus Egyetem, 2015.

[3] J. I. Toledo-Alvarado et al., "Automatic Building of an Ontology from a Corpus of Text Documents Using Data Mining Tools", 2012

[4] Joe Tekli, "An overview on XML Semantic Disambiguation from Unstructured", Member, IEEE, 2016.

[5] Harris, Z., 'The structure of science information', J Biomed. Inform., Vol. 35(4), pp. 215–221, 2002.

[6] Alexander Gelbukh, "Special issue: Natural Language Processing and its Applications", Institut Politécnico Nacional Centro de Investigaciónen Computación México, Mexico, 2010.

[7] Sibarani E. M., Nadial M., Panggabean E., & Meryana S., "A Study of parsing process on natural language processing in Bahasa Indonesia", International Conference on Computational Science and Engineering, pp. 309-316 2013.

[8] Andreas Hotho, Andreas Nürnberger, and Gerhard Paaß, "A Brief Survey of Text Mining. In Ldv Forum", Vol. 20.19–62. 2005.

[9] Dragomir R Radev, Eduard Hovy, and Kathleen McKeown, "Introduction to the special issue on summarization", Computational linguistics 28, 4, pp. 399–408, 2002.

[10] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut., Text Summarization Techniques: A Brief Survey. ArXiv e-prints, 2017, arXiv:1707.02268

[11] Dipanjan Das and André FT Martins, "A survey on automatic text summarization", Literature Survey for the Language and Statistics II course at CMU 4, pp. 192–195, 2007.

[12] Pritam C Gaigole, L. H. Patil, & P. M. Chaudhari, "Preprocessing Techniques in Text categorization", National Conference on Innovative Paradigms in Engineering & Technology (NVIPET-2013), Proceedings published by International Journal of Computer Applications (IJCA), 2013.

[13] Katariya Nikita, & Chaudhari M. S., "Text Preprocessing For Text Mining Using Side Information", International Journal of Computer Science and Mobile Applications, vol.3 Issue. 1, pp. 01-05, 2015.

[14] Ramasubramanian C., & Ramya R., "Effective Pre-Processing Activities in Text Mining using Improved Porter's Stemming Algorithm", International Journal of Advanced Research in Computer and Communication Engineering, vol. 2, Issue 12, pp. 4536-4538, 2013.

[15] Vijayarani S, Ilamathi J, & Nithya, International Journal of Computer Science & Communication Networks, Vol 5(1), pp. 7-16, 2015.

[16] Vijayarani S, & Janani R, "Text mining: open source tokenization tools–an analysis", Advanced Computational Intelligence 3.1: pp. 37-47, 2016.

[17] Vaidya, Swapnil, & Jayshree Aher, "Natural Language Processing Preprocessing Techniques", International Journal of Computer Engineering and Applications, Volume XI, Special Issue, 2017, www.ijcea.com ISSN 2321-3469

[18] Katariya Nikita, & Chaudhari M. S., "Text Preprocessing For Text Mining Using Side Information", International Journal of Computer Science and Mobile Applications, vol.3 Issue. 1, pp. 01-05, 2015.

[19] Nayak Arjun Srinivas, Kanive Ananthu, Chandavekar Naveen, & Balasubramani R, "Survey on Pre-Processing Techniques for Text Mining", International Journal Of Engineering And Computer Science, Volume 5 Issues 6 2016.

[20] Nazri Mohd Zakree Ahmad, Siti Mariyam Shamsudin, &Azuraliza Abu Bakar. "An exploratory study of the Malay text processing tools in ontology learning.", Research project, Ministry of Higher Learning – Malesia, 2008.

Authors Profile

Ravi Lourdusamy received his Ph.D from Bharathidasan University, Tamil Nadu, India in 2012. Currently he is HOD and Associate Professor in the Department of Computer Science, Sacred Heart College, Tirupattur, India. His research areas include Software Engineering, Ontology and Knowledge Engineering. He has presented and published over 20 papers in international conferences and journals. He was awarded a couple of UGC funded projects.

Stanislaus Abraham is a Research Scholar at the Department of Computer Science, Sacred Heart College, Tirupattur, India.

