CONCEPTUAL FRAMEWORK FOR ABSTRACTIVE TEXT SUMMARIZATION

Nikita Munot[1] and Sharvari S. Govilkar[2]

[1,2]Department of Computer Engineering, Mumbai University, PIIT, New Panvel, India

ABSTRACT

As the volume of information available on the Internet increases, there is a growing need for tools helping users to find, filter and manage these resources. While more and more textual information is available on-line, effective retrieval is difficult without proper indexing and summarization of the content. One of the possible solutions to this problem is abstractive text summarization. The idea is to propose a system that will accept single document as input in English and processes the input by building a rich semantic graph and then reducing this graph for generating the final summary.

KEYWORDS

Part-of speech (POS) tagging, rich semantic graph, abstractive summary, named entity recognition (NER).

1.INTRODUCTION

Text summarization is one of the most popular research areas today because of the problem of the information overloading available on the web, and has increased the necessity of the more strong and powerful text summarizers. The condensation of information from text is needed and this can be achieved by text summarization by reducing the length of the original text. Text summarization is commonly classified into two types extractive and abstractive. Extractive summarization means extracting few sentences from the original document based on some statistical factors and adding them into summary. Extractive summarization usually tends to sentence extraction rather than summarization. Whereas abstractive summarization are more powerful than extractive summarization because they generate the sentences based on their semantic meaning. Hence this leads to a meaningful summarization which is more accurate than extractive summaries.

Summarization by extractive just extracts the sentences from the original document and adds them to summary. Extractive method is based on statistical features not on semantic relation with sentences [2] and are easier to implement. Therefore the summary generated by this method tends to be inconsistent. Summarization by abstraction needs understanding of the original text and then generating the summary which is semantically related. It is difficult to compute abstractive summary because it needs understanding of complex natural language processing tasks.

There are few issues of extractive summarization. Extracted sentences usually tend to be longer than average. Due to this, parts of the segments that are not essential for summary also get included, consuming space. Important or relevant information is usually spread across sentences, and extractive summaries cannot capture this (unless the summary is long enough to hold all those sentences). Conflicting information may not be presented accurately. Pure extraction often leads to problems in overall coherence of the summary. These problems become more severe in the multi-document case, since extracts are drawn from different sources. Therefore abstractive

summarization is more accurate than extractive summarization.

In this paper, an approach is presented to generate an abstractive summary for the input document using a graph reduction technique. This paper proposes a system that accepts a document as input and processes the input by building a rich semantic graph and then reducing this graph for generating summary. Related work and literature survey is discussed in section 2. The proposed system is discussed in section 3 and conclusion in section 4.

2. LITERATURE SURVEY

In this section we cite the relevant past literature that use the various abstractive summarization techniques to summarize a document. Techniques till today focused on extractive summarization rather than abstractive. Current state of art is statistical methods for extractive summarization.

Pushpak Bhattacharyya [3] proposed a WordNet based approach to text summarization. It extracts a sub-graph from the WordNet graph for the entire document. Each nodes of the sub-graph are assigned weights with respect to the synsnet using the WordNet. WordNet[11] is a online lexical database. The proposed algorithm captures the global semantic information using WordNet.

Silber G.H., Kathleen F. McCoy [4][5] presents a linear time algorithm for lexical chain computation. Lexical chain is used as an intermediate representation for automatic text summarization. Lexical chains exploit the cohesion among an arbitrary number of related words. Lexical chains can be computed in a source document by grouping (chaining) sets of words that are semantically related. Words must be grouped such that it creates a strongest and longest lexical chain.

J. Leskovec[6] proposed approach which produces a logical form analysis for each sentence. The author proposed subject-predicate-object (SPO) triples from individual sentences to create a semantic graph of the original document. Difficult to compute SOP semantic based triples as it requires deep understanding of natural language processing.

Clustering is used to summarize a document by grouping and clustering the similar data or sentences. Zhang Pei-yin, LI zcun-he[7] states that summarization result depends on the sentence features and on the sentence similarity measure. MultiGen[7] is a multi-document system in the news domain.

Naresh Kumar, Dr.Shrish Verma[8] proposed a single document frequent terms based text summarization algorithm. The author suggests an algorithm based on three steps: First the document which is required to be summarized is processed by eliminating the stop word. Next step is to calculate the term-frequent data from the document and frequent terms are selected, and for these selected words the semantic equivalent terms are also generated. Finally in third step, all the sentences in document, which contains the frequent terms and semantic equivalent terms are filtered for summarization.

I. Fathy, D. Fadl, M. Aref[9] proposed a new semantic representation called Rich Semantic Graph(RSG) to be used as an intermediate representation for various applications. A new model to generate an English text from RSG is proposed. The method access a domain ontology which contains the information needed in same domain of RSG.

The author suggested a method [10] for summarizing document by creating a semantic graph and identifies the substructure of graph that can be used to extract sentences for a document summary.

It starts with deep syntactic analysis of the text. For each sentence it extracts logical form triples in the form of subject-predicate and object.

Many approaches addressed above uses lexical chain, word net and clustering method to produce abstractive summary. Some of the methods provided a graph-based approach to generate extractive summary.

3. PROPOSED APPROACH

The idea is to summarize an input document by creating semantic graph called rich semantic graph(RSG) for the original document, reducing the generated semantic graph, and the finally generating the final abstractive summary from the reduced semantic graph. The input to the system is a single text document in English and output will be a reduced summary.

The proposed approach includes three phases: Rich Semantic Graph creation (RSG) phase, Rich Semantic Graph reduction (RSG) phase and generating summary from reduced RSG.

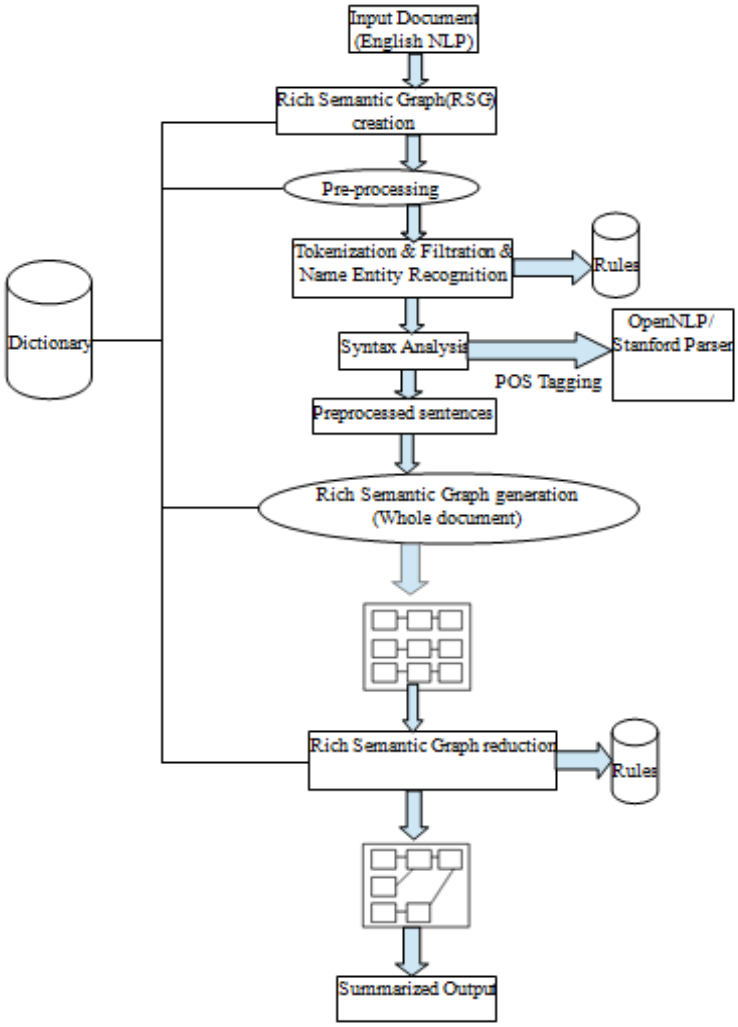

Figure 1. Proposed approach

First step is to pre-process the input document. For each word in the document, apply part-of-

speech tagging, named entity recognition and tokenization. Then for each sentence in the input document, graphs are created. Finally the sentences RSG sub graph are merged together to represent whole document semantically. The final RSG of entire document is reduced with the help of some reduction rules. Summary is generated from reduced RSG.

Algorithm:

Input: Accepts a single document as input.
Output: Summarized document.

Accept the text document as input in English
    for each sentence in the input document
        for each word in the sentences
            do tokenization
            part-of-speech tagging (POS)
            named entity recognition (NER)
    Generate the graph for each sentence
for entire document do
    merge all sentence graph to represent whole document
    reduce the graph using reduction rules
    generate from reduced graph

3.1. Rich Semantic Graph (RSG) Creation Phase

This phase analyses the input text, and detects the sentence and generates tokens for the entire document. For each word it generates POS tags and locates the words into predefined categories such as person name, location and organization. Then it builds the graph for each sentence and later it interconnects rich semantic sub-graphs. Finally the sub-graphs are merged together to represent the whole document semantically. RSG creation phase involves following tasks:

3.1.1. Pre-processing module

This phase accepts an input text document and generates pre-processed sentences. Initially the entire text document is taken as input. First step is to perform tokenization for document. Once tokens are generated, next step is to identify part-of-speech tag for every word or token and assign parts of speech to each word such as noun, verb, and adjective. These tags are useful for generating graph for entire document. Next task is to perform named entity recognition to identify the entities in the document. Pre-processing consists of 3 main processes: tokenization, parts of speech tagging (POS) and named entity recognition (NER)[1]. Once tokens, part-of-speech (POS), Named Entity Recognition (NER) are ready these tags are used for further generating graph of each sentence.

Steps for pre-processing module [1]:

1. Tokenization & Filtration: Accept the input document, detect sentences and generate the tokens for entire document and filter out the special characters.

2. Name Entity Recognition (NER): It locates atomic elements into predefined categories such as location, person names, organization etc.To perform this task we have used Stanford NER tool [15] which is available freely.

3. Part of speech tagging (POS): It parses the whole sentence to describe each word syntactic function and generates the POS tags for each word. To perform this task, Stanford parser tool [12] is used for implementation.

Algorithm for pre-processing module:

Input: Original input document.
Output: Tokens, POS tags & NER.

accept the single text document as input
generate tokens for entire document and store in a file
for each sentence, apply POS tagging and generate the POS tags for each words in sentences
for each sentence, locate the atomic elements into predefined categories such as person, organization etc and identify the proper nouns
apply sentence detection algorithm to generate the sentence in proper order.

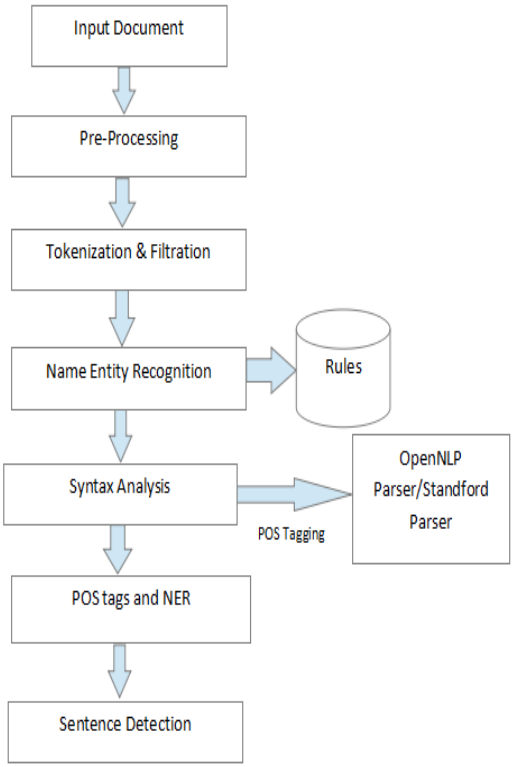

Figure 2. Pre- Processing module

This phase accepts a input document and filters special character and unwanted script other than English. Then it generates tokens, Name Entity Recognition (NER) and part-of-speech (POS) tags for all the sentences.

Tokenization & Filtration:

Tokenization is a process of breaking a stream of sentences into tokens. This is done by searching a space after each word. All the generated tokens are saved in a separate file for further processing. In this phase the input text is filtered so as to remove all the special characters such as * &^%$ #@,.';+{}[]. Including special characters in the further processing will result in only degradation of the performance. Also filtration of Devanagari script(Marathi, Hindi) is also done to validate that input document is in English language only.

Algorithm:

1. Generate a list of all the possible special characters.
2. Then compare each character of input text file with a given list of special characters.
3. If match founds then we simply ignore the matched character.
4. If no match found then that character is not a special character, simply copy the character into another file containing no special characters.
5. Repeat the step from 2 to 4 till every character from the input text get processed.
6. Give generated file with no special characters to next step
7. Stop.

Input:

!@$%#&*()&^^%%+_?><": **प्रतापगडाची लढाई** ही इतिहासातील महत्वाची Alice Mathew is a graduate student. Alice lives in Mumbai. Bob John is a graduate student. Bob works in Mastek.

Output:

Alice
Mathew
is
a
graduate
student
Alice
lives
in
Mumbai
Bob
John
is
a
graduate
student

After tokenization & filtration all the special characters will be removed and non English words will be removed and tokens will be generated and saved in a separate file.

Named Entity Recognition (NER):

Named Entity Recognition (NER) labels sequences of words in a text which are the names of

things, such as person and company names, or organization and location. Good named entity recognizers for English, particularly for the 3 classes (PERSON, ORGANIZATION, and LOCATION) are available. There are tools available for performing these tasks such as Stanford NER tool and Open NLP tool. Consider the following sentence and expected named entity tags are identified by using Stanford NER tool [15]:

Input:

Alice Mathew is a graduate student. Alice lives in Mumbai. Bob John is a graduate student. Bob works in Mastek.

Output:

Alice Mathew	Person
Mumbai	Location
Bob John	Person
Mastek	Organization

POS tagging

A Part-of-Speech Tagger (POS Tagger) is a piece of software that reads text in some language and assigns parts of speech to each word, such as noun, verb, adjective, etc. OpenNLP POS tagging tool and Stanford parser tool [12] is available that can be used as a plug-in. The OpenNLP POS Tagger uses a probability model to predict the correct POS tag out of the tag set. Penn Treebank POS tag set [13][14] is available which are used by many applications. The proposed method used Stanford parser tool [12] for part-of-speech tagging.

Input:

Alice Mathew is a graduate student. Alice lives in Mumbai. Bob John is a graduate student. Bob works in Mastek.

Output:

Alice_NNP Mathew_NNP is_VBZ a_DT graduate_NN student._NN Alice_NNP lives_VBZ in_IN Mumbai_NNP. Bob_NNP John_NNP is_VBZ a_DT graduate_NN student._NN. Bob_NNP works_VBZ in_IN Mastek_NNP.

3.1.2. Rich Semantic sub-graph generation module

This module accepts the pre-processed sentences as input and generates graph for each sentence and later the sub-graphs are merged together to represent the entire document. For every sentence graphs are generated. The semantic graph is generated with the help of generated POS tags and tokens where the noun are coloured in orange and verbs in red color in the form of subject-predicate-object(SPO) triples.

Input:

Alice Mathew is a graduate student. Alice lives in Mumbai.

Output:

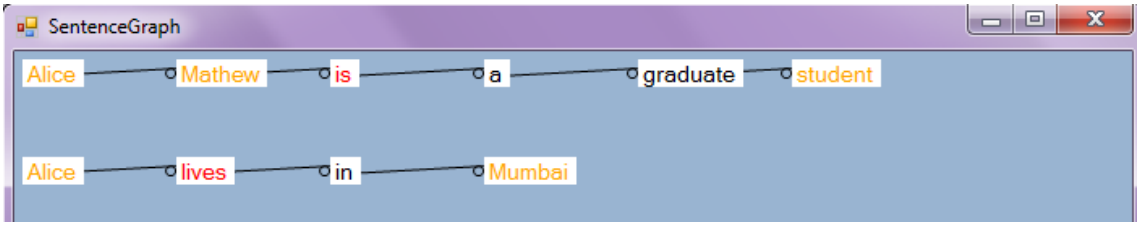

Figure 3. Sentence graph

3.1.3. Rich Semantic Graph Generation module

Graph theory [6] can be applied for representing the structure of the text as well as the relationship between sentences of the document. Sentences in the document are represented as nodes. The edges between nodes are considered as connections between sentences. These connections are related by similarity relation. By developing different similarity criteria, the similarity between two sentences is calculated and each sentence is scored. Whenever a summary is to be processed all the sentences with the highest scored are chosen for the summary. In graph ranking algorithms, the importance of a vertex within the graph is iteratively computed from the entire graph. Therefore the graphs can be generated for the entire document.

The rich semantic graph generation module is responsible to generate the final rich semantic graphs of the whole document. The semantic sub-graphs are merged together to form the final RSG. The graph can be built by subject-predicate-object (SPO) triples from individual sentences to create a semantic graph. It uses linguistic properties of the nodes in the triples to build semantic graphs for both documents and corresponding summaries.

Extracting summary by semantic graph generation [10] is a method which uses subject-predicate-object (SPO) triples from individual sentences to create a semantic graph of the original document. Subjects, Objects, and Predicates are the main functional elements of sentences. Identifying and exploiting links among them could facilitate the extraction of relevant text. A method that creates a semantic graph of a document, based on logical form triples subject–predicate–object (SPO), and learns a relevant sub-graph that could be used for creating summaries.

The semantic graph is generated in two steps [10]:

1. Syntactic analysis of the text – First apply deep syntactic analysis to document sentences , and extract logical form triples.

2. Finally merge the resulting logical form triples into a semantic graph and analyze the graph properties. The nodes in graphs correspond to Subjects, objects and predicate.

Input: Pre-processed sentences as input (POS tags and NER).
Output: Rich Semantic Graph.

Consider the following input and graph generated for same.

Input:

Rini lives in Mumbai. She works in Infosys. Nikita is pursuing master degree in computer engineering. Nikita is specialized in machine learning field. Rini John is a graduate student completed engineering in computer science. She is specialized in Web NLP. Rini is also pursuing post graduation in computer science. Rini is a friend of Nikita. Ashish Mathew is also friend of Nikita. Nikita Munot published two papers in international conferences under guidance of Prof.Sharvari Govilkar. Rini John also published two papers in international conferences under guidance of Prof.Sharvari Govilkar.

Output:

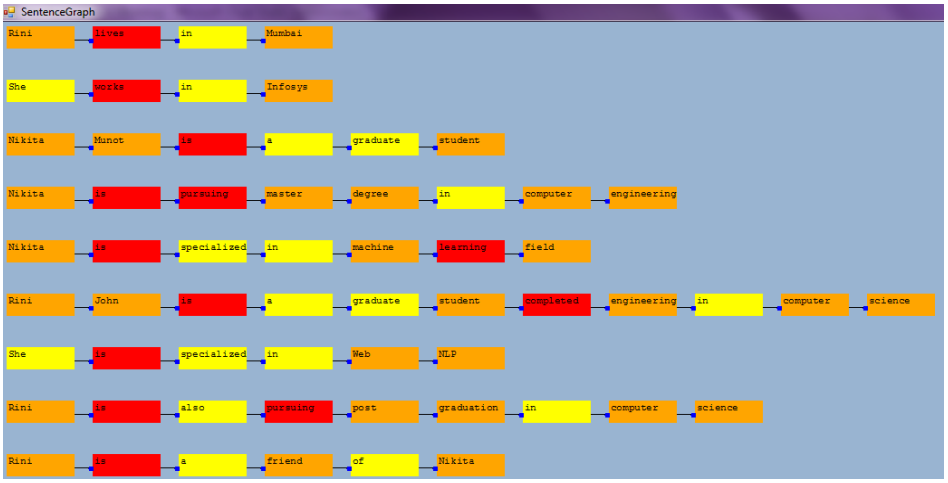

Figure 4. Rich semantic graph

The graphs are generated in subject-predicate-object form where the noun are coloured orange and verbs in red and proper noun in orange.

3.2. Rich Semantic Graph Reduction Phase

This phase reduces the generated rich semantic graph of the original document. In this process a set of rules are applied on the generated RSG to reduce it by merging, deleting or consolidating the graph nodes. Many rules can be derived based on many factors: the semantic relation, the graph node type (noun or verb), the similarity or dissimilarity between graph nodes.

Few rules are discussed that can be applied on the graph nodes of two simple sentences:

Sentence1= [SN1, MV1, ON1]
Sentence2= [SN2, MV2, ON2]

Each sentence is composed of three nodes: Subject Noun (SN) node, Main verb (MV) node and Object Noun (ON) node.

Input: Rich Semantic Graph (RSG) of the whole document.

Output: Reduced rich semantic graph (RSG).

Reduction rules examples [1]:

Rule 1. IF	SN1 is instance of noun N	And
	SN2 is instance of noun N	And
	MV1 is similar to MV2	And
	ON1 is similar to ON2	
THEN		
	Merge both MV1 and MV2	And
	Merge both ON1 and ON2	
Rule 2. IF	SN1 is instance of subclass of noun N	And
	SN2 is instance of subclass of noun N	And
	{[MV11, ON11],..[MV1n, ON1n]}	is similar to
	{[MV21, ON21],..[MV2n, ON2n]}	
THEN		
	Replace SN1 by N1 (instance N)	And
	Replace SN2 by N2 (instance N)	And
	Merge both N1 and N2	
Rule 3. IF	SN1 and SN2 are instance of noun N	And
	MV1 is instance of subclass of verb V	And
	MV2 is instance of subclass of verb V	And
	ON1 is similar to ON2	
THEN		
	Replace MV1 by V1 (instance V)	And
	Replace MV2 by V2 (instance V)	And
	Merge both V1 and V2	And
	Merge both ON1 and ON2	

With the help of such rules, the graph is reduced then final summary is generated from reduced graph. The system is to be trained and more such rules are to be added to make the system more strong.

3.3 Summarized Text Generation Phase

This phase aims to generate the abstractive summary from the reduced RSG. The sentences are merging with the help of rules and final summary can be generated.

4. CONCLUSION

As natural language understanding improves, computers will be able to learn from the information online and apply what they learned in the real world. Information condensation is needed. Extractive summary leads usually for sentence extraction rather the summarization. So the need is to generate summary that captures the important text and relates the sentences semantically. The work is applicable in open domain.

Abstractive summarization will serve as a tool for generating summary which is semantically correct and produced fewer amounts of sentences in summary. Extractive summarization leads to sentence extraction based on statistical methods which are not useful always. This paper proposes an idea to create a semantic graph for the original document and relate it semantically and by using several rules reduce the graph and generate the summary from reduced graph.

REFERENCES

[1] Ibrahim F. Moawad, Mostafa Aref," Semantic Graph Reduction Approach for Abstractive Text Summarization" 2012 IEEE

[2] Saeedeh Gholamrezazadeh, Mohsen Amini Salehi, Bahareh Gholamzadeh, "A Comprehensive Survey on Text Summarization Systems" 2009 In proceeding of: Computer Science and its Applications, 2nd International Conference.

[3] Kedar Bellare, Anish Das Sharma, Atish Das Sharma, Navneet Loiwal and Pushpak BhattachaIbrahim F.Moawadryya, "Generic Text Summarization Using Wordnet", Language Resources Engineering Conference (LREC 2004), Barcelona, May, 2004.

[4] Silber G.H., Kathleen F. McCoy, "Efficiently Computed Lexical Chains as an Intermediate Representation for Automatic Text Summarization," Computational Linguistics 28(4): 487-496, 2002.

[5] Barzilay, R., Elhadad, M, "Using Lexical Chains for Text Summarization," in Proc. ACL/EACL'97 Workshop on Intelligent Scalable Text summarization, Madrid, Spain,1997, pp.10–17.

[6] J. Leskovec, M. Grobelnik, N. Milic-Frayling, "Extracting Summary Sentences Based on the Document Semantic Graph", Microsoft Research, 2005.

[7] Pei-ying, LI Cun-he," Automatic text summarization based on sentences clustering and extraction", 2009.

[8] Naresh Kumar, Dr.Shrish Verma , "A Frequent Term And Semantic Similarity Based Single Document Text Summarization Algorithm" 2011.

[9] I. Fathy, D. Fadl, M. Aref, "Rich Semantic Representation Based Approach for Text Generation", The 8th International conference on Informatics and systems (INFOS2012), Egypt, 2012.

[10] J. Leskovec, M. Grobelnik, N. Milic-Frayling, "Learning Semantic Graph Mapping for Document Summarization", 2000.

[11] C. Fellbaum, "WordNet: An Electronic Lexical Database", MIT Press,1998.

[12] Stanford Parser,http://nlp.stanford.edu:8080/parser/index.jsp, June 15, 2012.

[13] R. Prasad, N. Dinesh, A. Lee, E. Miltsakaki, L. Robaldo, A. Joshi, B. Webber, "The Penn Discourse Treebank 2.0", Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008), Morocco.

[14]https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html.

[15] http://nlp.stanford.edu/software/CRFNER.html.

Authors

Nikita Munot received B.E. degree in 2012 from Pillai's Institute of Information Technology, New Panvel,Mumbai University and currently pursuing M.E. from Mumbai university. She is having 3 years teaching experience. Presently she is working as a lecturer in Pillai's Institute of Information Technology. Her research areas include natural language processing and data mining. She has published one paper in international journal.

Sharvari Govilkar is Associate professor in Computer Engineering Department, at PIIT, New Panvel, University of Mumbai, India. She has received her M.E in Computer Engineering from University of Mumbai. Currently She is pursuing her PhD in Information Technology from University of Mumbai. She is having seventeen years of experience in teaching. Her areas of interest are Text Mining, Natural language processing, Information Retrieval & Compiler Design etc. She has published many research papers in international and national journals and conferences.

.