Review on Natural Language Processing Trends and Techniques Using NLTK

Deepa Yogish[1], T. N. Manjunath[2](✉), and Ravindra S. Hegadi[3](✉)

[1] VTU-RC-ISE, BMSIT&M, Bangalore, India
deepayogish@gmail.com
[2] ISE, BMSIT&M, Bangalore, India
manju.tn@bmsit.in
[3] School of Computational Sciences, Solapur University, Solapur, India
rshegadi@sus.ac.in

Abstract. In modern age of information explosion, every day millions of gigabytes of data are generated in the form of documents, web pages, e-mail, social media text, blogs etc., so importance of effective and efficient Natural Language Processing techniques become crucial for an information retrieval system, text summarization, sentiment analysis, information extraction, named entity recognition, relationship extraction, social media monitoring, text mining, language translation program, and question answering system. Natural Language Processing is a computational technique applies different levels of linguistic analysis for representing natural language into a useful representation for further processing. NLP is recognized as a challenging task in computer science and artificial intelligence because understanding human natural language is not only depends on the words but how those words are linked together to form precise meaning is also considered. Regardless of language being one of the easiest concepts for human to learn, but for training computers to understand natural language is a difficult task due to the ambiguity of language syntax and semantics. Natural Language processing techniques involves processing documents or text which reduces storage space and also reduces the size of index and understanding the given information which satisfies user's need. NLP techniques improve the performance of the information retrieval efficiency and effective documentation processes. Common dialect handling procedures incorporates tokenization, stop word expulsion, stemming, lemmatization, parts of discourse labeling, lumping and named substance recognizer which enhances execution of NLP applications. The Natural Language Toolkit is the best possible solution for learning the ropes of NLP domain. NLTK, a collection of application packages which encourage researchers and learners in natural language processing, computational linguistics and artificial intelligence.

Keywords: Natural Language Processing (NLP) · Artificial Intelligence (AI) · Information Retrieval (IR) · Natural Language Tool Kit (NLTK)

1 Introduction

Natural Language Processing is a territory of software engineering, man-made consciousness, and computational semantics. NLP is related to human-computer interaction leveraged many applications such as document summarization, sentiment analysis, topic modeling, named entity recognition, relationship extraction, text mining, language translation, and question answering system. Natural language processing is influenced by many organizations to improve the effective documentation processes and identify the most relevant information from large data corpus. Many organizations use NLP techniques to optimize customer support by collecting the useful information. NLP improvises the efficiency of text analytics and enhance social media monitoring. For example, banks might implement NLP algorithms to optimize customer support and a large consumer products brand might combine natural language processing and semantic analysis to improve their knowledge management strategies and social media monitoring. Natural Language Processing is a computational technique applies different levels of linguistic analysis for representing natural language into a useful representation for further processing. NLP is recognized as a challenging problem in computer science and artificial intelligence because understanding human natural language is not only depends on the words but how those words are linked together to form precise meaning is also considered. Regardless of language being one of the easiest concepts for humans to learn, but for training computers to understand natural language is a difficult task due to the ambiguity of language.

Developers utilize Natural Language Processing concepts and techniques to organize and structure the data or information to perform different NLP applications which includes:

- **Machine Translation:** Machine translation program which automatically translate text from one form to another form where human and machine can understand. Google has a translation program used as statistical engine which translates each word in a sentence to other readable form by preserving the meaning of each word in a sentence.
- **Email Spam Filtering:** Spam filters have become important as the protection against unwanted email from unknown persons. The issues of spam filters are at the challenging task of NLP technology to extract correct meaning from strings of text in email to avoid spam emails.
- **Information Extraction:** The task of NLP is to provide pertinent information by extracting important entities which are affected by news, opinions and announcements in the form of natural language for financial market decisions.
- **Summarization:** NLP task in summarization is to produce a short summary which contains pertinent information from large amounts of data such as to produce a short summary from large academic articles. Summarization is used to understand customer opinion based on collective data from social media from which a company can decide next product offers.
- **Question Answering system:** Challenging task of NLP in question answering system is to understand Natural language queries and to provide short and

precise answer from large data corpus. Google's efforts in NLP in answering user question has focused in providing relevant documents but still Question Answering system remains a major challenge for search engines to provide exact answer to the user queries.

- **Social Media Monitoring:** Web based life checking gives an extraordinary chance to organizations to recognize what their customers are discussing via web-based networking media stages, web journals, and so forth and to find pertinent data for their future business choices. By communicating with customers, preparing their discussions and basically understanding clients in their very own words, organizations can more likely comprehend their customers' needs and enhance the associations with them.
- **Text Analytics:** Many associations use characteristic dialect preparing to approach content issues and enhance exercises, for example, information administration and huge information examination [12]. Morphological, linguistic [13], syntactic and semantic examinations of dialect empower distinguishing proof and extraction of elements like themes, areas, individuals, association, dates, and so forth and produce the metadata that can be utilized to tag and classify content in the most exact way.

Components of NLP

Natural Language Understanding (NLU) is to understand chunks of text data in the given natural language and to preprocess into a user understandable representation using different level of analysis.

Natural Language Generation (NLG) is to generate text in to human readable language with meaningful phrases and sentences in the natural language using different level of synthesis.

General Steps in Natural Language Processing
Figure 1 depicts the stages of analysis [1] in processing natural language with different level of analysis such as morphological analysis, syntactic analysis, semantic analysis, discourse analysis and pragmatic analysis.

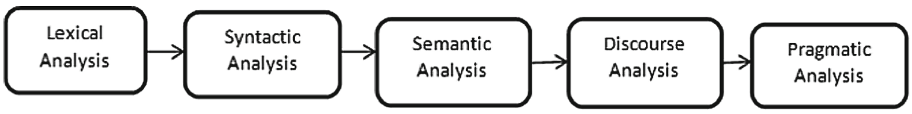


Fig. 1. Stages of analysis in processing natural language

- **Lexical Analysis:** Lexical Analysis involves analyzing the computational meaning of each word in context of document and divides the whole text into paragraphs, sentences and words.
- **Syntactic Analysis:** Syntactic analysis involves analysis of the rules or grammar of the sentence which will give a correct meaning by providing an

order and structure of each sentence in the text. Syntactic analysis is responsible for creation of a syntactic structure through parse tree of the sentence. Syntactic analyzer rejects the sentence if they violate the rules of language. Ex- "Blue dress girl the go the to market".

- **Semantic Analysis:** Semantic analysis means finding out the dictionary meaning of the text for meaningfulness. Sentence is rejected if no semantic meaning. Ex- "colorless - blue dress girl go to the market" meaningless sentence.
- **Discourse Analysis:** To frame remedy importance in a sentence which relies upon the sentence that goes before it and furthermore impacted by the significance of sentences that tail it. Ex-"Blue dress young lady required it" "it" relies on young lady.
- **Pragmatic Analysis:** Pragmatic investigation gets learning from outside presence of mind data and gives intentional utilization of dialect in circumstance. Ex-"Do you recognize what time she goes to showcase?" Interpreted as a demand.

2 Standard NLP Toolkit

Few toolkits are supporting for NLP tasks for the development of many applications. The current challenge is to select the best tool depends on application specific, source of text and performance of NLP tasks. Natural Language Toolkit (NLTK) is a gathering of program modules, libraries and diverse content change procedures for English dialect written in Python programming dialect. NLTK was produced by Steven Bird and Edward Loper at the University of Pennsylvania. The NLTK toolbox is in charge of various NLP assignments, for example, tokenization, stemming, parse tree portrayals, labeling, parsing, chunking and NER. NLTK stuffed with test codes for corpus readers, tokenizers, stemmers, taggers, chunkers, parsers, word net and NER. NLTK utilizes the Penn treebank tokenizer and POS tagger utilizes the Penn treebank tagset. The ACE corpus with a Maximum Entropy prototype is utilized to prepare chunkers and NER modules.

Apache OpenNLP toolkit is a Java library for natural language tasks. Features of Apache OpenNLP include tokenization, POS tagging, NER, chunking, sentence segmentation, co-reference resolution and parsing. The POS tagging and chunking for English language uses the Penn treebank tags and CoNLL-2000 dataset is used to train chunker.

Stanford CoreNLP toolbox is a Java library for normal dialect handling assignments. Stanford CoreNLP is in charge of numerous NLP assignments, for example, labeling, named entity recognizer [NER], co reference goals, assumption examination, bootstrapped design learning and data extraction.

The CoreNLP utilizes a Penn Treebank style tokenization; POS module utilizes Penn Treebank tagset with usage of the Maximum Entropy demonstrate. Conditional Random Field (CRF) display is utilized for NER utilizing CoNLL-2003 dataset.

Pattern is a Python library underpins NLP, web mining and Machine Learning undertakings. Provides modules to Google, Twitter, Wikipedia API, a web crawler, POS labeling, estimation investigation, Word Net, Vector space model, grouping and SVM.

GATE (General Architecture for Text Engineering) created by division of Computer Science, University of Sheffield with License GNU LGPL. Door bolsters NLP undertakings with program modules, for example, tokenizer, sentence splitter, POS tagger, NER and co-reference goals tagger.

TextBlob is a Python library like NLTK gives modules to NLP errands, for example, POS labeling, tokenization, Chunking, assumption investigation, order, interpretation, and some more.

SpaCy is a modern quality programming library written in Python and Cython for cutting edge Natural Language Processing applications. Backings tokenization, NER, POS labeling, parsing and word vectors.

Table 1 shows comparing different NLP toolkits [2] for NLP techniques like tokenization, POS tagging, chunking, NER tagging and stemming. NLTK, OpenNLP and CoreNLP can perform almost all lower-level NLP tasks. We have chosen the NLTK as best tool for research and applications to best of our knowledge. NLTK is the best solution for learning the ropes of NLP domain. Its modular structure helps comprehend the dependencies between components and get the best experience with composing appropriate models for solving certain tasks.

Table 1. Comparison of various NLP toolkits

NLP toolkit	Programming language	Tokenization	POS tagging	Chunking	NER	Stemmer
NLTK	Python	Yes	Yes	Yes	Yes	Yes
Apache OpenNLP	Java	Yes	Yes	Yes	Yes	Yes
Stanford CoreNLP	Java	Yes	Yes	No	Yes	Yes
Pattern	Python	Yes	Yes	Yes	No	No
GATE	Java	Yes	Yes	Yes	Yes	No
Spacy	Python/Cython	Yes	Yes	Yes	Yes	No

3 Natural Language Processing Techniques Using NLTK

NLTK [3] is written in Python which supports many features like transparent syntax, good string-handling features and easy to understand. Python programming as an object oriented language supports interactive interpreter, encapsulated data and code which has extensive library. NLTK uses the treebank word tokenizer and POS tagger utilizes the Penn Treebank tagset which is prepared on the PENN treebank corpus with a Maximum Entropy prototype. Chunking and the NER modules are utilized on ACE corpus with a Maximum Entropy prototype.

3.1 Text Transformation Techniques Using NLTK

NLTK is a gathering of test Python modules for some NLP tasks, for example, tokenization, stop word expulsion, stemming, POS tagging,chunking and NER. Brown Corpus, CoNLL-2000 Chunking Corpus, CMU Pronunciation Dictionary, NIST IEER Corpus, PP Attachment Corpus, Penn Treebank, and the SIL Shoebox corpus design are the corpus tests incorporated into NLTK.

Tokenization. Tokenization is a procedure of producing significant tokens from textual information. Tokenization is a process of identification of tokens within documents which reduces storage space and satisfies user's information need more precisely with reduced search space. Janani [4] investigated the execution of seven open source tokenization tools. For tokenize an archive, numerous tools are accessible. The tools are Nlpdotnet Tokenizer, Mila Tokenizer, NLTK Word Tokenizer, TextBlob Word Tokenizer, MBSP Word Tokenizer, Pattern Word Tokenizer, and Word Tokenization with Python NLTK. Among all above tokenization tools Word Tokenization with Python NLTK is the best tool which gives the best result. This tool can take up to 50000 characters at a time. The text is first tokenized into sentences using the PunktSentence Tokenizer. Then each sentence is tokenized into words using four different word tokenizers such as TreebankWordTokenizer, WordPunctTokenizer, PunctWordTokenizer and WhitespaceTokenizer. NLTK supports treebankWordTokenizer which uses regular expressions to tokenize text with Penn Treebank corpus. The text will be segmented into sentences by using method sent_tokenize ( ) and word_tokenize ( ) method is invoked for tokenization.

Stop Word Removal. The most commonly used preprocessing method in NLP application is stop word removal. The main purpose of stop word removal is removing the words that occur commonly across all the documents in the corpus. A generally utilized stop words are "the", "an", "an", "in". Stop-word removal [5] is utilized to enhance the execution of the Information Retrieval System, Text Analytics, Text Summarization and Question-Answering framework. NLTK import stop words and set stop word for English dialect by executing command on python shell as demonstrated as follows. import nltk from nltk.corpus import stopwords set(stopwords.words("English")).

Stemming. Stemming is a most commonly used preprocessing technique in text mining and Information retrieval systems. Stemming process used by information retrieval systems for indexing words by reducing the size of index and search space by increasing retrieval accuracy. Stemming [6] process reduces different grammatical word forms to its root form. The "stem" is acquired by applying an arrangement of standards without thinking about the grammatical feature (POS) with regards to the word event in sentence. Over stemming and under stemming [6] are two blunders in stemming. Over stemming happens when two words with various stems are stemmed to indistinguishable root and known from a false positive. Under-stemming happens when two words that ought to be stemmed to a similar root are not and known as a false negative. The issue of over stemming and under stemming can be decreased by thinking about the language structure, semantics and parts of discourse of a sentence. Figure 2 depicts classification of stemming algorithms for stemming process.

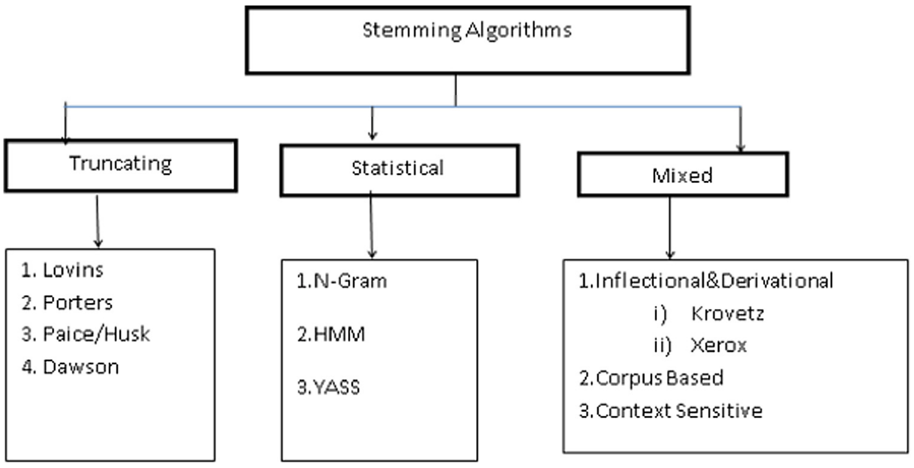


Fig. 2. Classification of stemming algorithms

Truncating Methods (Affix Removal): Truncating methods produces stem by removing affixes of a word based on the some value 'n' that is to consider word till 'n' letters and to discard rest of all alphabets.

Statistical Methods: Statistical stemmers are based on statistical analysis and techniques produces stem by applying few statistical procedure.

Mixed Methods: stemmers in hybrid methods depend on both the inflectional and derivational morphology analysis with a given corpus to develop these types of stemmers.

The Porter's Stemmer is the most widely used by researchers; the basic algorithm will be changed based on their requirements [6]. The Porter stemmer was produced by Martin Porter in 1980 at the University of Cambridge. The Porter stemmer [7] is a setting touchy postfix expulsion calculation comprises of number of direct advances, five or six utilized to create the last stem. The control looks like <condition> <suffix> leads to <new suffix>.

POS Tagging. Part of Speech labeling (POS) is a method of allotting proper grammatical feature tag for each word in a given content. Label set is the arrangement of labels from which the tagger chooses proper descriptor for each word. The coarse-grained label set are N (Noun), V (Verb), ADJ (descriptive word), ADV (Adverb), PREP (Preposition), CONJ (Conjunction) and fine-grained label set are NNOM (Noun-Nominative), NSOC (NounSociative), VFIN (Verb limited), VNFIN (Verb Nonfinite). Generally most taggers utilize just fine grained label set. POS labeling [9] is a normally utilized pre-preparing module for different NLP undertakings like Machine interpretation, Natural dialect content handling, synopsis, Multilingual and cross dialect data recovery, Speech acknowledgment, Artificial knowledge, Parsing, Expert framework. Difficulties of POS labeling, for example, remote words, ambiguities, ungrammatical information and so forth. The uncertainty in POS labeling can be fathomed utilizing the sentence structure rules.

POS Tagging Process. Part of Speech labeling (POS) is a procedure of relegating a proper grammatical feature labels utilizing standard label set for each word in information content. Figure 3 describes POS tagging process in which tagging algorithm assign tags for each token comparing with specified tag set.

Order of POS labeling is delineated in Fig. 4 which named supervised strategy and unsupervised method. Supervised POS taggers [10] depend on pre-labeled corpora, which increment the execution of the models with the expansion in size of pre-labeled corpora. Unsupervised POS labeling models don't require pre-labeled corpora however utilize computational strategies to appoint naturally labels to words [10].

Rule-based part-of-speech tagging uses hand-written rules based on morphological and contextual information. Disambiguation is settled by examining the lexical, semantic, syntax of each word in a sentence, its previous word, its following word and different perspectives. For instance, if the former word is article then the word in question must be a noun. This information is coded in the form of hand written rules in rule based tagger. The rule tagger [8] process is divided into two stages. In first stage, tagger returns a set of possible POS tags by searching words in dictionary and in second stage uses a set of hand-coded rules to assign a single part of speech for each word by removing disambiguate words using linguistic features of word. Examples of rule based tagger are TAGGIT and ENGTWOL. A stochastic approach assigns a tag to word using probability and statistics. Stochastic approaches assign a tag based on the probability of certain tag occurrences in a word sequence and predictions based on past observations. Example of stochastic tagger is CLAWS (constituent likelihood automatic word -tagging system). The hybrid approach [8] is a combination of Rule based approach and statistical approach. Hybrid tagger assign most probable tag to the word using statistical method, if any disambiguate found after assigning tag, then apply grammar rules to change. Hybrid tagger uses rule-based systems to specify tags using hand

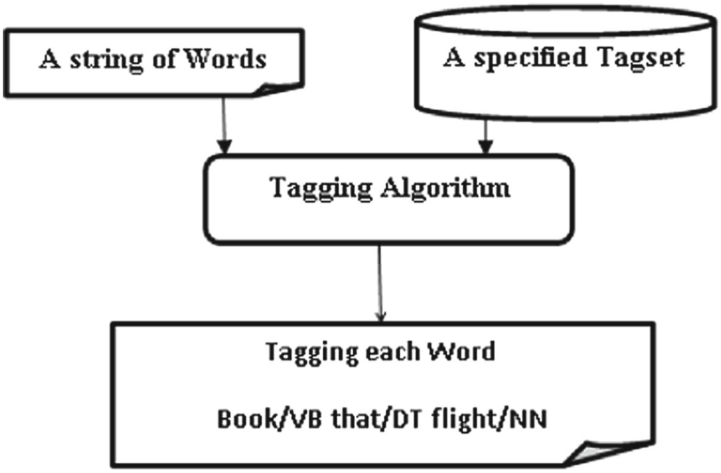


Fig. 3. POS tagging process

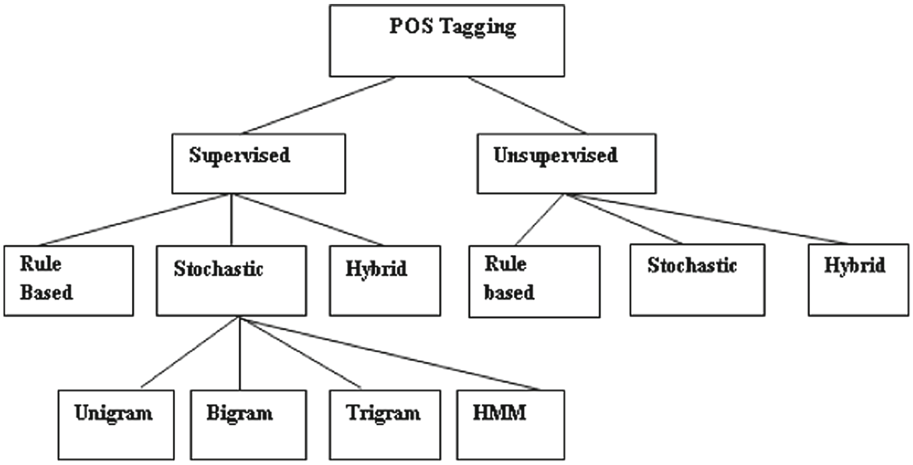


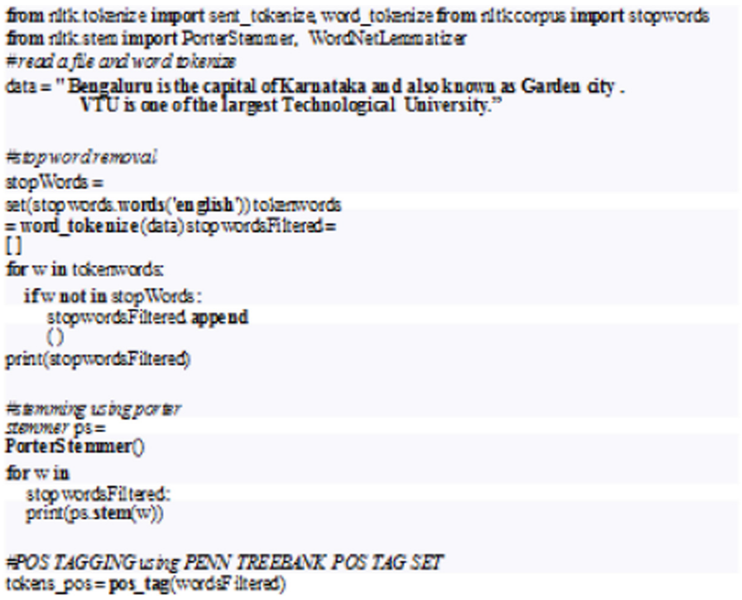
Fig. 4. Classification of POS tagging

written rules and use stochastic systems by applying machine-learning to induce rules from a tagged training corpus automatically. Example of hybrid approach is Brill tagger or transformation based tagger. NLTK POS tagger utilizes the Penn Treebank label set as default and is prepared on the PENN Treebank corpus with a Maximum Entropy prototype. Table 2. Demonstrates the Penn Treebank POS label set.

Table 2. Penn TreeBank POS tag set

Tag	Description	Example	Tag	Description	Example
CC	Coordinating Conjunction	For, and, while,but, or	SYM	Symbol	+, %, &
CD	Cardinal number	one, five, ten	TO	Infinitive 'to'	to go
DT	Determiner	a, the, some	UH	Interjection	ah, oops
EX	Existential There	There	VB	Verb, base form	Get, write, jump
FW	Foreign word	mea culpa	VBD	Verb, past tense	Got
IN	Preposition/sub-conjunction	With, of, in, by	VBG	Verb, gerund	Getting
JJ	Adjective	Yellow, pretty, old	VBN	Verb, past participle	Eaten
JJR	Adj., comparative	Bigger	VBP	Verb, non 3sg present	Get
JJS	Adj., superlative	Wildest, biggest	VBZ	Verb, 3sg present	Gets
LS	List item marker	1, 2, one	WDT	Wh-determiner	Which, that
MD	Modal	Can, should	WP	Wh-pronoun	What, who
NN	Noun, sing, or mass	Rama, dog	WPS	Possessive wh	Whose
NNS	Proper noun, singular	Ramas	WRB	Wh-adverb	How, where
NNPS	Proper noun, plural	Computers	$	Dollar sign	$
PDT	Predeterminer	Both the boys	#	Pound sign	#
POS	Possessive ending	Friend's	"	Left quote	(")
PP	Personal pronoun	I, you, she	"	Right quote	(")
PRP$	Possessive pronoun	your, his	{	Left parameters	(—, (,{,<)
RB	Adverb	quickly, never	}	Right parameters	(—,),},>)
RBR	Adverb, comparative	Faster	,	Comma	,
RBS	Adverb, superlative	Fastest	.	Sentence-final punc	(. ! ?)
RP	Particle	up, off	:	Mid-sentence punc	(: : ... - -)

Figure 5 shows code snippet for tokenization, stop word removal, stemming and POS tagging process using word_ tokenize ( ) method, importing stop words and setting for English language, using porter stemmer algorithm which instantiate the Porter Stemmer class and call the stem ( ) method using Penn Treebank tagset.



```
from nltk.tokenize import sent_tokenize, word_tokenize from nltk.corpus import stopwords
from nltk.stem import PorterStemmer,  WordNetLemmatizer
#read a file and word tokenize
data = "Bengaluru is the capital of Karnataka and also known as Garden city .
          VTU is one of the largest Technological  University."

#stop word removal
stopWords =
set(stopwords.words('english')) tokenwords
= word_tokenize(data) stopwordsFiltered =
[]
for w in tokenwords:
  if w not in stopWords :
     stopwordsFiltered.append
     ()
print(stopwordsFiltered)

#stemming using porter
stemmer ps =
PorterStemmer()
for w in
  stopwordsFiltered:
  print(ps.stem(w))

#POS TAGGING using PENN TREEBANK POS TAG SET
tokens_pos = pos_tag(wordsFiltered)
```

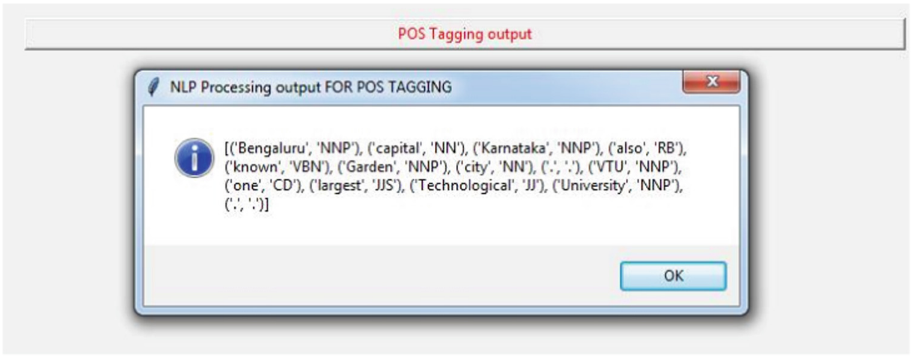
Fig. 5. Code snippet for tokenization, stop word removal, stemming and POS tagging process

Figure 6 shows snapshot of tokenization, stop word removal, stemming and POS tagging process.

Lemmatization. Lemmatization is the way toward decreasing a gathering of words into their lemma. Lemmatization delivers the "lemma" of a word after fathom the Parts of Speech and the setting of a word in the given sentence. Lemmatization used in information retrieval application like indexing and searching as a normalization technique. Riddhi et al. [11], discussed five different Lemmatization approaches such as Edit Distance on dictionary algorithm, Morphological analyzer, radix trie, Affix lemmatizer and fixed length truncation approach. Figure 7 shows Lemmatization process instantiate WordNetLemmatizer class using lemmatize ( ) method.

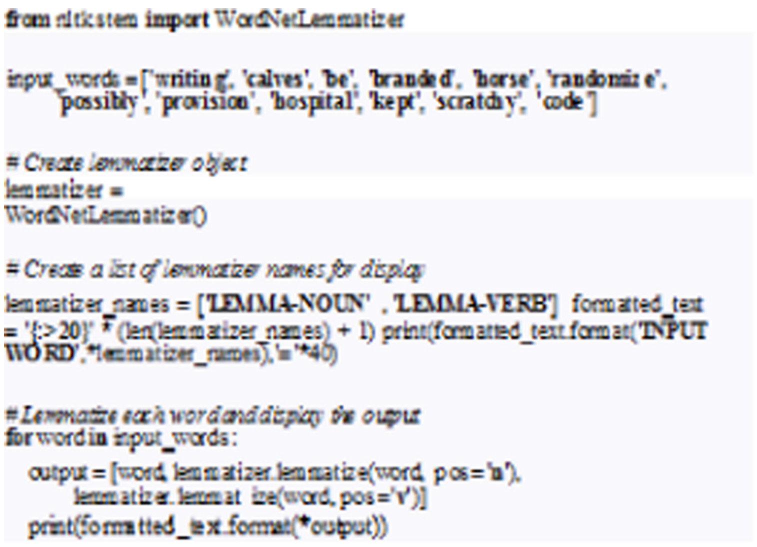
Figure 8 shows snapshot for Lemmatization process using lemmatize ( ) method for noun and verb parts of speech for different inputs.

Word Frequency – Frequency Distribution. After processing text using tokenizion, stop word removal and stemming we can figure the word recurrence that is checking the occasions every token happens in the information corpus. Python object Counter is used to perform word frequency.



```
[('Bengaluru', 'NNP'), ('capital', 'NN'), ('Karnataka', 'NNP'), ('also', 'RB'),
('known', 'VBN'), ('Garden', 'NNP'), ('city', 'NN'), ('.', '.'), ('VTU', 'NNP'),
('one', 'CD'), ('largest', 'JJS'), ('Technological', 'JJ'), ('University', 'NNP'),
('.', '.')]
```

Fig. 6. Snapshot of tokenization, stop word removal, stemming and POS tagging process

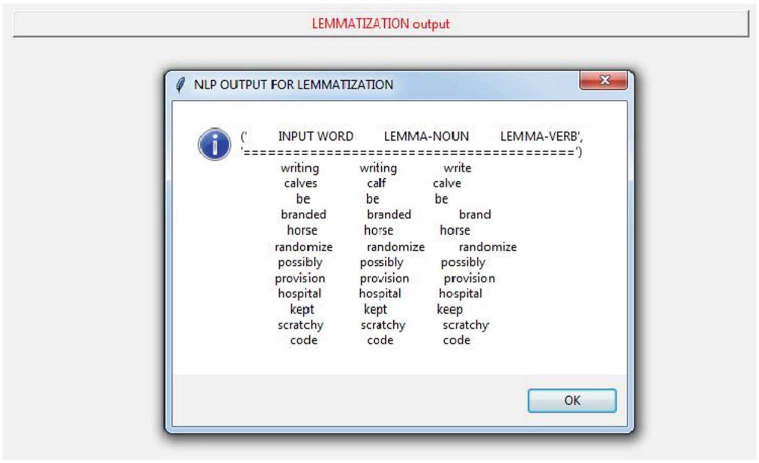


```
from nltk.stem import WordNetLemmatizer

input_words =['writing', 'calves', 'be', 'branded', 'horse', 'randomize',
      'possibly', 'provision', 'hospital', 'kept', 'scratchy', 'code']

# Create lemmatizer object
lemmatizer =
WordNetLemmatizer()

# Create a list of lemmatizer names for display
lemmatizer_names = ['LEMMA-NOUN' , 'LEMMA-VERB']  formatted_text
= '{:>20}' * (len(lemmatizer_names) + 1) print(formatted_text.format('INPUT
WORD',*lemmatizer_names),'='*40)

# Lemmatize each word and display the output
for word in input_words:
   output = [word, lemmatizer.lemmatize(word, pos='n'),
        lemmatizer.lemmat ize(word, pos='v')]
   print(formatted_text.format(*output))
```

Fig. 7. Code snippet for Lemmatization process

Figure 9 shows code snippet for counting frequency of each token in document using counter ( ) method after preprocessing given text using tokenization, stop word removal and stemming.

Figure 10 shows snapshot for counting frequency of each token in document using counter ( ) method.



LEMMATIZATION output

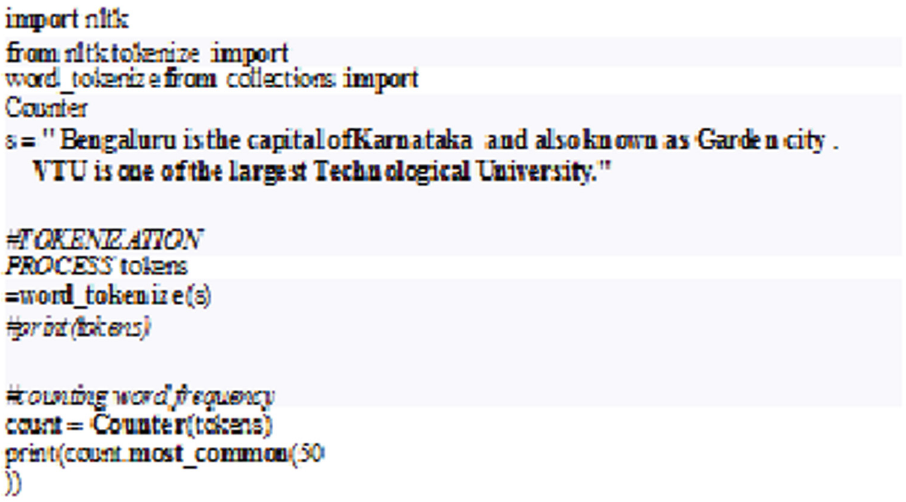
NLP OUTPUT FOR LEMMATIZATION

```
('     INPUT WORD      LEMMA-NOUN       LEMMA-VERB',
'========================================')
   writing        writing        write
   calves          calf         calve
     be             be           be
   branded        branded          brand
    horse          horse        horse
  randomize        randomize       randomize
  possibly       possibly        possibly
  provision      provision        provision
  hospital       hospital        hospital
     kept           kept         keep
  scratchy       scratchy         scratchy
     code           code          code
```

OK

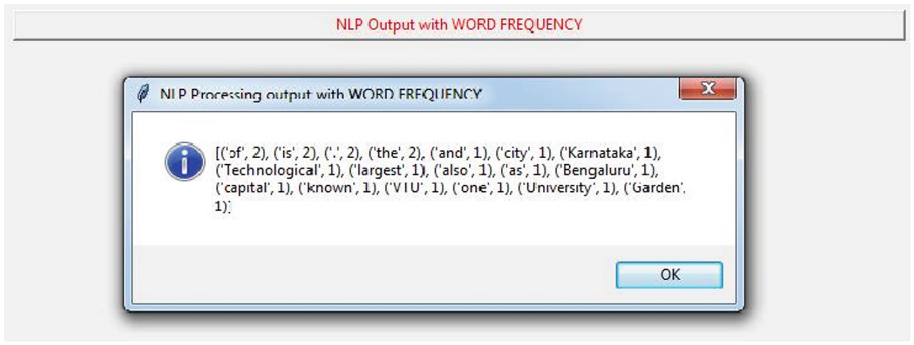
Fig. 8. Snapshot of Lemmatization process for noun and verb POS

Named Entity Recognizer (NER). Named-Entity Recognition (NER) is an essentially connected in errand of data extraction for extricating basic substances, for example, the people, areas, associations, articulations of times, amounts, fiscal qualities, rates, and so on. Named Entity Recognition is a type of content mining that isolates out unstructured content information and finds thing phrases called named elements. Question Answering (QA) framework utilizes NER method to enhance the exactness of Information Retrieval by recovering important archive which contains a response to the client's inquiry.



```
import nltk
from nltk.tokenize import
word_tokenize from collections import
Counter
s = "Bengaluru is the capital of Karnataka  and also known as Garden city .
   VTU is one of the largest Technological University."

#TOKENIZATION
PROCESS tokens
=word_tokenize(s)
#print(tokens)

#counting word frequency
count = Counter(tokens)
print(count.most_common(50
))
```

Fig. 9. Code snippet for word frequency using counter ( ) method


NLP Output with WORD FREQUENCY
NLP Processing output with WORD FREQUENCY
OK

```
[('of', 2), ('is', 2), ('.', 2), ('the', 2), ('and', 1), ('city', 1), ('Karnataka', 1),
('Technological', 1), ('largest', 1), ('also', 1), ('as', 1), ('Bengaluru', 1),
('capital', 1), ('known', 1), ('VTU', 1), ('one', 1), ('University', 1), ('Garden',
1)]
```

Fig. 10. Snapshot for word frequency using counter ( ) method

NER is performed after pre-processing of records by tokenization, stop word expulsion and Part-Of-Speech(POS) Tagging. Named Entity Recognition is the assignment of recognizing substances in a sentence with labels like a man as PERSON, association as ORG, date as DATE, area as GPE, time as TIME and so on. Figure 11 shows code snippet for named entity recognizer using nltk.ne_chunk( ) function.

Figure 12 shows snapshot of named entity recognizer representing tags as person, location, organization for a given text.

Chunking. Chunking is a procedure of extricating noun or verb phrases from unstructured content. Chunking is applied after POS tagging and provides chunks as output. Chunking utilizes standard arrangement of Chunk labels like Noun Phrase (NP), Verb Phrase (VP), and Adjective Phrase (AP) and so forth. Chunking is essentially utilized when we need to remove data from content, for example, Locations, Person Names and so forth. Chunking is a three stage process, for example, labeling a sentence by POS tagger, Chunk the POS labeled sentence and break down the parse tree to extricate data. In order to create an NP or VP chunker, define a chunk grammar consisting of rules with a single regular expression. In order to chunk, combine the part of speech tags with regular expressions. To shape standard articulation, use meta characters, for example, + = at least one or more events of the former component.

? = zero or one events of the former component.

* = zero or More events of the former component.

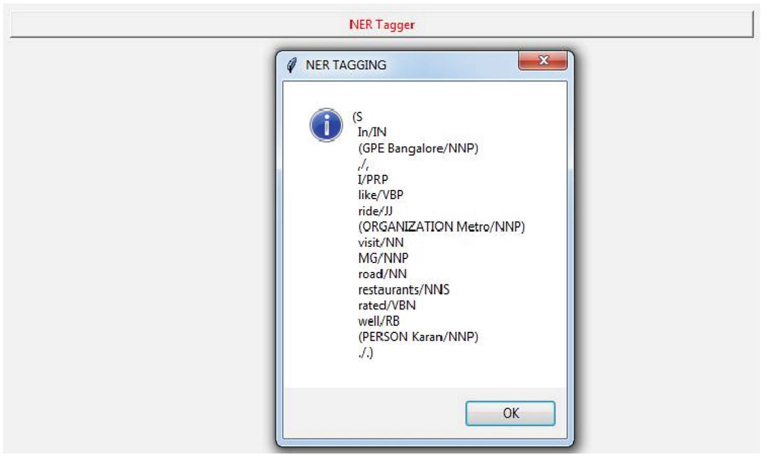
. = Any character with the exception of another line.

The part of speech tags are denoted with the "<" and ">" and place regular expressions within the tags to chunk the tagged sentence. Few examples are shown below.

<JJ.?>* = "0 or more of any tense of adjective", <VB.?>* = "0 or more of any tense of verb", <NNP>+ = "One or more proper nouns".

```
import nltk
from nltk tokenize import sent_tokenize,
word_tokenize from nltk corpus import stopwords
from nltk stem import PorterStemmer,
WordNetLemmatizer from nltk import pos_tag

data = "In Bangalore, I like to ride the Metro to visit MG road and some restaurants rated well by Karan."
#print (word_tokenize(data))

#stop word removal
stop Words =
set(stopwords.words('english'))
tokenwords = word_tokenize(data)
stop wordsFiltered = [ ]

for w in tokenwords:
  if w not in stop Words :
     stopwordsFiltered.appen
     d(w)
print(stop wordsFiltered)

#stemming using porter
stemmer ps =
PorterStemmer()
for w in
  stopwordsFiltered:
  print(ps.stem(w))

#POS TAGGING using PENN TREEBANK POS TAG SET
tokens_pos = pos_tag(wordsFiltered)

#Named entity recognizer
print(nltk.ne_chunk(tokens_p
os))
```

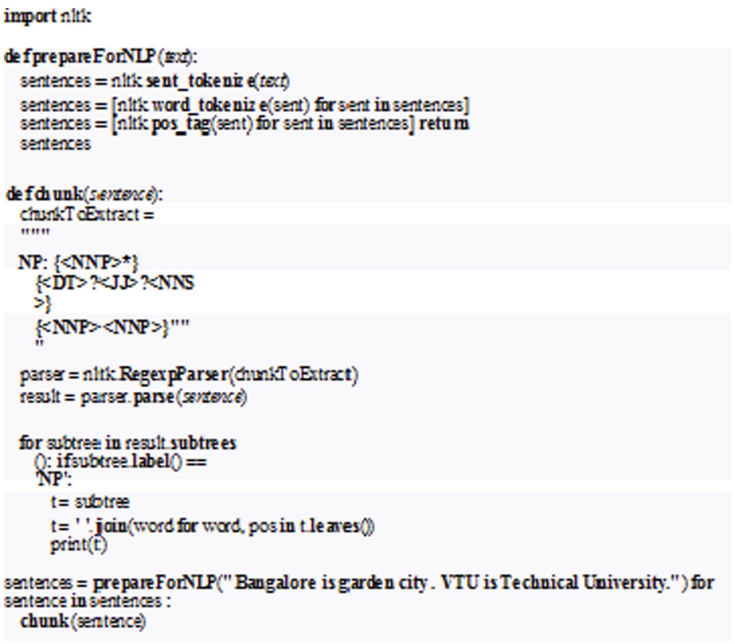
Fig. 11. Code snippet for NER process



```
(S
 In/IN
 (GPE Bangalore/NNP)
 ,/,
 I/PRP
 like/VBP
 ride/JJ
 (ORGANIZATION Metro/NNP)
 visit/NN
 MG/NNP
 road/NN
 restaurants/NNS
 rated/VBN
 well/RB
 (PERSON Karan/NNP)
 ./.)
```

Fig. 12. Snapshot of Named Entity recognizer

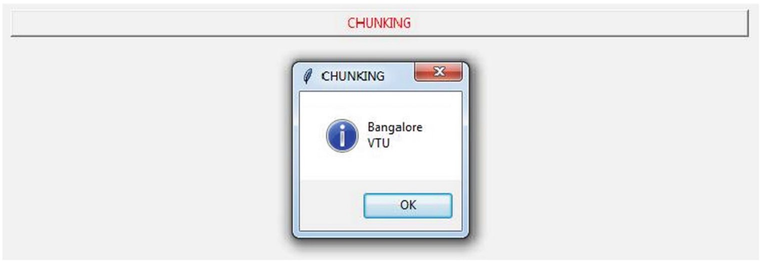
Figure 13 illustrates chunking process using regular expression. For a given sentence, preprocess the text using tokenization and POS tagging. Chunk the tagged text using regular expression to noun phrase, verb phrase or adverb phrase.



```
import nltk

def prepareForNLP(text):
  sentences = nltk.sent_tokenize(text)
  sentences = [nltk.word_tokenize(sent) for sent in sentences]
  sentences = [nltk.pos_tag(sent) for sent in sentences] return
  sentences

def chunk(sentence):
  chunkToExtract =
  """

 NP: {<NNP>*}
    {<DT>?<JJ>?<NNS
    >}
    {<NNP><NNP>}""
    "

  parser = nltk.RegexpParser(chunkToExtract)
  result = parser.parse(sentence)

  for subtree in result.subtrees
    (): if subtree.label() ==
    'NP':
      t= subtree
      t= ' '.join(word for word, pos in t.leaves())
      print(t)

sentences = prepareForNLP("Bangalore is garden city. VTU is Technical University.") for
sentence in sentences :
  chunk(sentence)
```

Fig. 13. Code snippet for Chunking process to chunk noun phrase

Figure 14 shows snapshot of chunking process to chunk noun phrase for a given sentence.



```
Bangalore
VTU
```

Fig. 14. Snapshot of chunking process for noun phrase

4 Conclusion

Huge volume of data is generated every minute through social media platforms which are used by much organization for their business operations. In connection with decision making, making the machine to learn to arrive suitable decisions at the right time. In this regard Natural Language processing plays a vital role in machine learning. Language being one of the easiest things for humans to learn, but for training computers to understand natural language is a difficult problem due to the ambiguity of language syntax and semantics. Information retrieval process provides the world's wealth of information at our fingertips, but still required improvement and challenging task to provide precise answer for answering specific questions posed by humans. Natural language processing is the process of transforming a natural language text into a structured format that a computer can process and produce useful information to users. So pre-processing of natural language is necessary for many applications. We have used NLTK tool kit for natural language preprocessing techniques such as tokenization, stop word removal, stemming, Lemmatization, POS tagging, NER tagging and chunking which reduces search space, reduce index size and improve the accuracy of retrieval process in many applications.

References

1. Swapnil, V., Jayshree, A.: Natural language processing preprocessing techniques. Int. J. Comput. Eng. Appl. **XI**(Special Issue) (2017). http://www.ijcea.com/. ISSN 2321-3469
2. Alexandre, P., Hugo, G.O., Ana, O.A.: Comparing the performance of Different NLP toolkits in formal and social media text. In: 5th Symposium on Languages, Applications and Technologies, Germany (2016).https://doi.org/10.4230/OASIcs, SLATE.2016
3. Steven, B.: NLTK: the natural language toolkit. In: Proceedings of the COLING/ACL Interactive Presentation Sessions, Association for Computational Linguistics, Sydney, pp. 69–72 (2006)
4. Vijayarani, S., Janani, R.: Text mining: open source tokenization tools - an analysis. Adv. Comput. Intell. Int. J. (ACII) **3**(1), 37–47 (2016)
5. Raulji, J.K., Saini, J.R.: Stop-word removal algorithm and its implementation for Sanskrit language. Int. J. Comput. Appl. (0975–8887) **150**(2), 15–17 (2016)
6. Jivani, A.G.: A comparative study of stemming algorithms. Int. J. Comp. Tech. Appl. **2**(6), 1930–1938 (2011). ISSN 2229-6093
7. Anjali, M.K., BabuAnto, P.: Parts of speech taggers for dravidian languages. Int. J. Eng. Trends Technol. (IJETT) **21**(7), 342–347 (2015). https://doi.org/10.14445/22315381/IJETT-V21P263. ISSN 2231-5381
8. Shubhangi, R., Sharvari, G.: Survey of various POS tagging techniques for Indian regional languages. Int. J. Comput. Sci. Inf. Technol. (IJCSIT) **6**(3), 2525–2529 (2015)
9. Mahar, J.A., Qadir, G.: MEMON: rule based part of speech tagging of Sindhi language. In: Proceeding of International Conference on Signal Acquisition and Processing (2010)

10. Mahar, J.A., Memon, G.Q.: Parts of speech taggers for Dravidian languages. Int. J. Eng. Trends Technol. (IJETT) **21**(7), 1933–1938 (2015). ISSN 2231-5381
11. Riddhi, D., Prem, B.: Survey paper of different lemmatization approaches. Int. J. Res. Advent Technol. (2015). (E-ISSN 2321-9637) Special Issue 1st International Conference on Advent Trends in Engineering, Science an d Technology
12. Manjunath, T., Ravindra, N., Hegadi, S.: Statistical data quality model for data migration business enterprise. Int. J. Soft Comput. Medwell J. (2013). ISSN 1816-9503
13. Ruikar, D.D., Hegadi, R.S.: Simple DFA construction algorithm using divide-and-conquer approach. In: Nagabhushan, P., Guru, D.S., Shekar, B.H., Kumar, Y.H.S. (eds.) Data Analytics and Learning. LNNS, vol. 43, pp. 245–255. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-2514-4_21