Vivekananda Institute of Technology

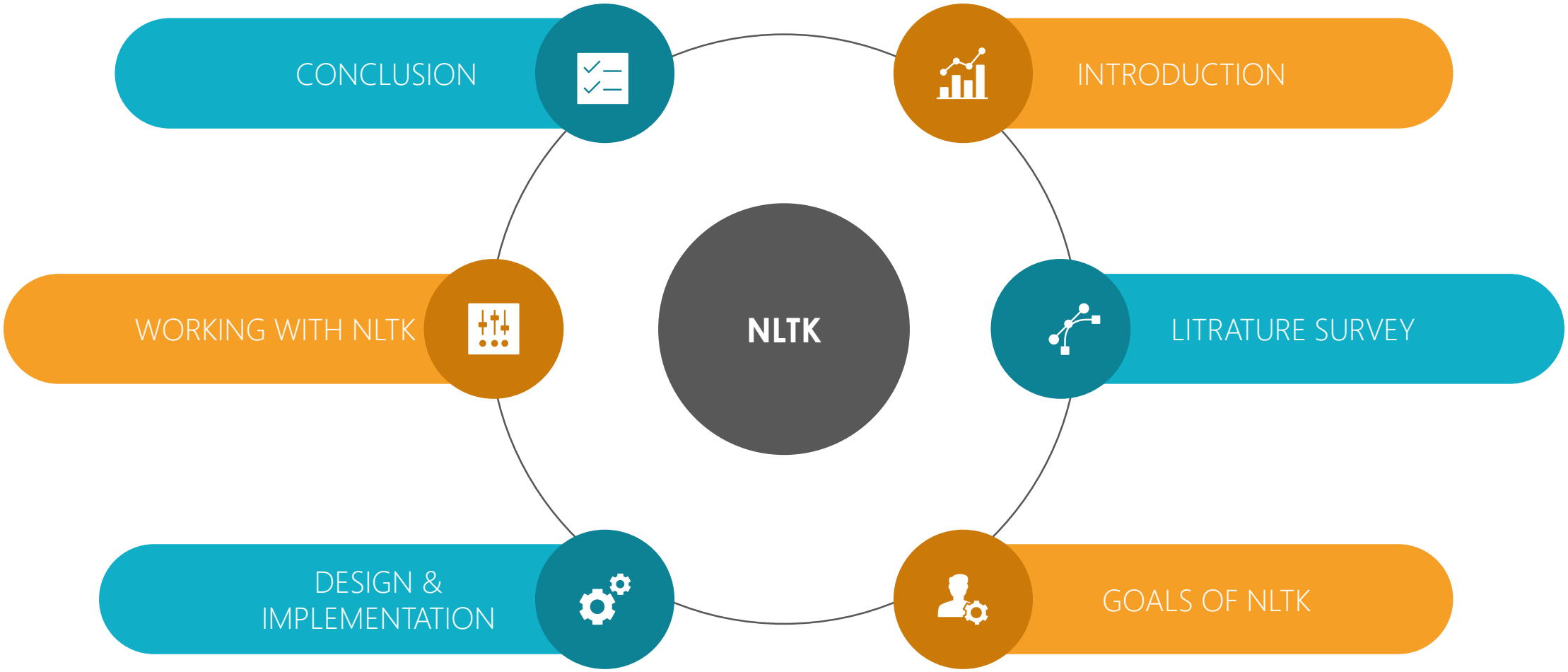
Technical Seminar

NATURAL LANGUAGE TOOL KIT

Name : Naveen Pandurangi
USN : 1VK15CS034
Subject Code : 15CS86

Seminar Guide
Mrs. Chandramma R
(Assoc. Professor, Dept. of CSE, VKIT, Bengaluru)

Contents


CONCLUSION
INTRODUCTION
WORKING WITH NLTK
NLTK
LITRATURE SURVEY
DESIGN & IMPLEMENTATION
GOALS OF NLTK

NLTK : Natural Language Tool Kit

- A set of Python modules to carry out many natural language tasks
- Basic classes to represent data for NLP
- Infrastructure to build NLP programs in Python
- Python interface to over 50 corpora and lexical resources
- Focus on machine learning with specific domain Knowledge
- Free and open source

Literature Survey

Natural Language Processing using NLTK and WordNet[1]

- It provides easy-to-use interfaces to many corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning.

Using NLTK for educational and scientific purposes [3]

- how necessary the NLTK is for the course of Computational Linguistics at the university and for researchers in the field of natural language processing.
  - high readability
  - an easy to use object-oriented paradigm
  - easy extensibility
  - strong Unicode support
  - a powerful standard library

NLTK: The Natural Language Toolkit (2009) [2]

- Modification to design
  - Tokens and other core data types
  - The corpus module
  - Processing modules

NLTK: The Natural Language Toolkit (2002) [4]

- NLTK was developed as a open source program modules
  - Parsing Modules
  - Tagging Modules
  - Finite State Automata
  - Type Checking
  - Visualization
  - Text Classification

GOALS OF NLTK

SIMPLICITY

To provide an intuitive framework along with substantial building blocks, giving users a practical knowledge of NLP without getting bogged down in the tedious house-keeping usually associated with processing annotated language data.

CONSISTANCY

To provide a uniform framework with consistent interfaces and data structures, and easily-guessable method names.

EXTENSIBILITY

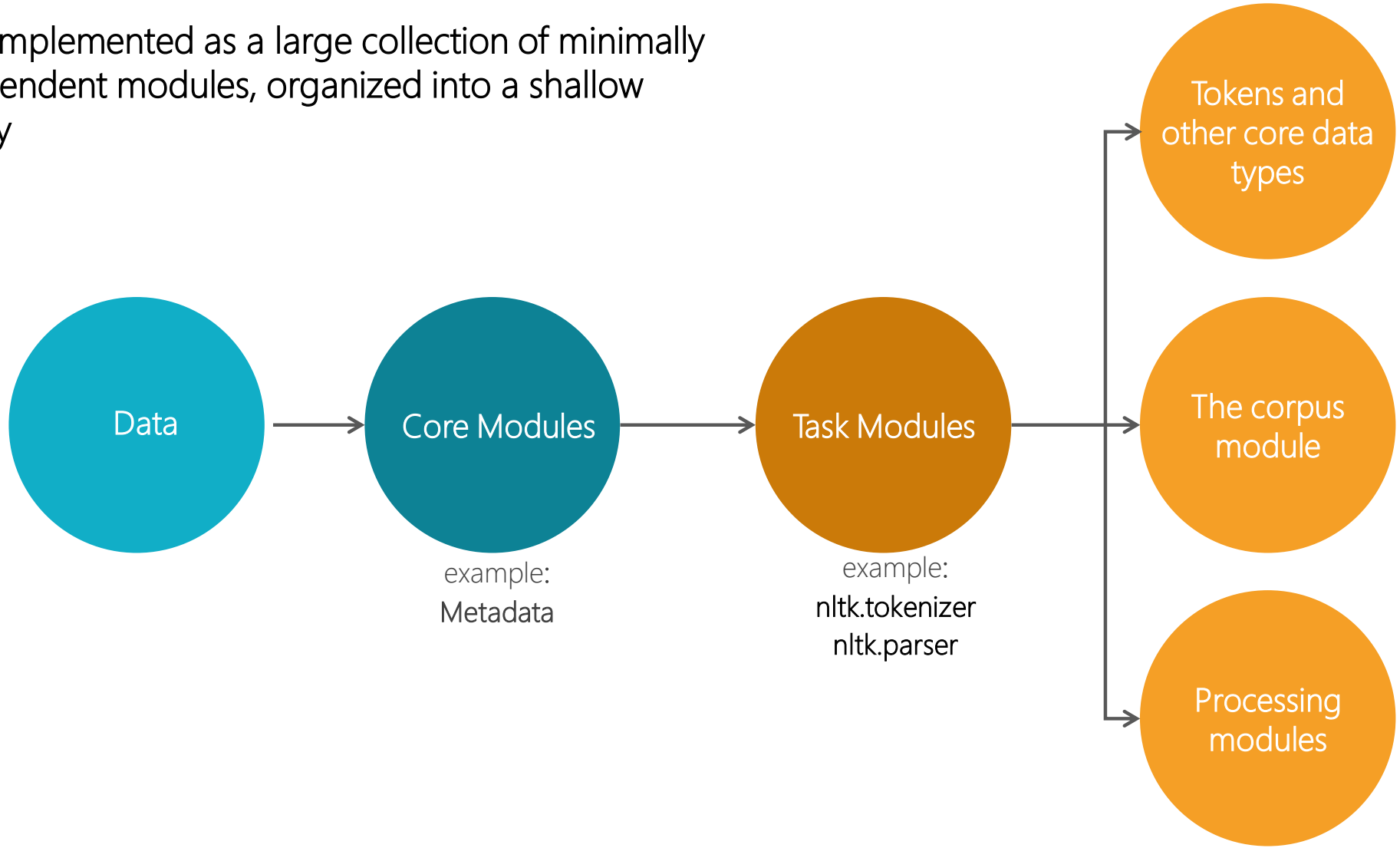
To provide a structure into which new software modules can be easily accommodated, including alternative implementations and competing approaches to the same task

MODULARITY

To provide components that can be used independently without needing to understand the rest of the toolkit.

Design & Implementation

- NLTK is implemented as a large collection of minimally interdependent modules, organized into a shallow hierarchy



Tokens and other core data types

- To maximize interoperability between modules a single class is encode information about natural language texts – the **Token** class
- the TEXT property is used to encode a token's text content:

```
>>> from nltk.token import *
>>> Token(TEXT="Hello World!")
<Hello World!>
```

- The TAG property is used to encode a token's part of-speech tag:

```
>>> Token(TEXT="python", TAG="NN")
<PYTHON/NN>
```

- The SUBTOKENS property is used to store a tokenized text:

```
>>> from nltk.tokenizer import *
>>> tok = Token(TEXT="Hello World!")
>>> WhitespaceTokenizer().tokenize(tok)
>>> print tok['SUBTOKENS'])
[<Hello>,<World>]
```

The corpus module

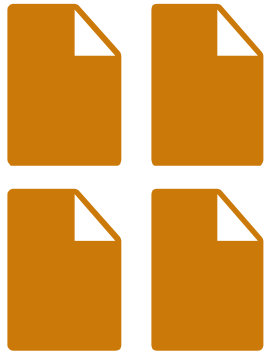
- The corpus module defines classes for reading and processing many of the corpora.

```
>>> from nltk.corpus import brown
>>> brown.groups()
['skill and hobbies', 'popular lore',
'humor', 'fiction: mystery', ...]
>>> brown.items('humor')
('cr01', 'cr02', 'cr03', 'cr04', 'cr05',
'cr06', 'cr07', 'cr08', 'cr09')
>>> brown.tokenize('cr01')
<[<It/pps>, <was/bedz>, <among/in>,
<these/dts>, <that/cs>, <Hinkle/np>,
<identified/vbd>, <a/at>, ...]>
```

```
>>> from nltk.corpus import treebank
>>> treebank.groups()
('raw', 'tagged', 'parsed', 'merged')
>>> treebank.items('parsed')
['wsj_0001.prd', 'wsj_0002.prd', ...]
>>> item = 'parsed/wsj_0001.prd'
>>> sentences = treebank.tokenize(item)
>>> for sent in sentences['SUBTOKENS']:
...     print sent.pp() # pretty-print
(S:
  (NP-SBJ:
    (NP: <Pierre> <Vinken>)
    (ADJP:
      (NP: <61> <years>)
      <old>
    )
    ...
```

Processing modules

- Each language processing algorithm is implemented as a class and NLTK includes the following modules:
  1. Cfg
  2. corpus
  3. draw (cfg, chart, corpus, featurestruct, fsa, graph, plot, rdparser, srparser, tree)
  4. eval
  5. featurestruct
  6. parser (chart, chunk, probabilistic)
  7. probability
  8. sense
  9. set
  10. stemmer (porter)
  11. tagger
  12. test
  13. token
  14. tokenizer
  15. tree
  16. util



Modules

- Token - classes for representing and processing individual elements of text, such as words and sentences
- Probability - classes for representing and processing probabilistic information
- Tree - classes for representing and processing hierarchical information over text
- Cfg - classes for representing and processing context free grammars

Modules

- Tagger - tagging each word with a part-of-speech, a sense, etc
- Parser - building trees over text (includes chart, chunk and probabilistic parsers)
- Classifier - classify text into categories (includes feature, featureSelection, maxent, naivebayes)
- Draw - visualize NLP structures and processes
- Corpus - access (tagged) corpus data

Tokenization

- Simplest way to represent a text is with a single string
- Difficult to process text in this format
- Convenient to work with a list of tokens
- Task of converting a text from a single string to a list of tokens is known as tokenization
- The most basic natural language processing technique
- Example - Word Tokenization

  Input : "Hey there, How are you all?"

  Output : "Hey", "there,", "How", "are", "you", "all?"

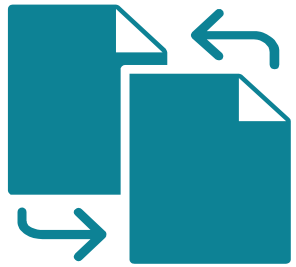
Text Corpus

- Large collection of text
- Concentrate on a topic or open domain
- May be raw text or annotated / categorized

Corpuses

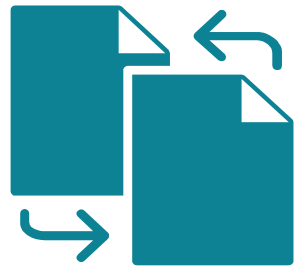
- Gutenberg - selection of e-books from Project Gutenberg
- Webtext - forum discussions, reviews, movie script
- nps_chat - anonymized chats
- Brown - 1 million word corpus, categorized by genre
- Reuters - news corpus
- Inaugural - inaugural addresses of presidents
- Udhr - multilingual corpus

Accessing Corpora



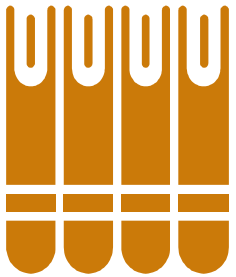
- Corpora on disk - text files
- NLTK provides Python modules / functions / classes that allow for accessing the corpora in a convenient way
- It is quite an effort to write functions that read in a corpus especially when it comes with annotations
- The task of reading in a corpus is needed in many NLP projects

Accessing Corpora



- # tell Python we want to use the Gutenberg corpus
- from nltk.corpus import gutenberg
- # which files are in this corpus?
- print(gutenberg.fileids())
- >>> ['austen-emma.txt', 'austen-persuasion.txt', 'austen-sense.txt', 'bible-kjv.txt', ...]

WordNet



- Structured, semantically oriented English dictionary
- Synonyms, antonyms, hyponims, hypernims, depth of a synset, trees, entailments, etc.

```
>>> from nltk.corpus import wordnet as wn
>>> wn.synsets('motorcar')  [Synset('car.n.01')]
>>> wn.synset('car.n.01').lemma_names
['car', 'auto', 'automobile', 'machine', 'motorcar']
>>> wn.synset('car.n.01').definition
'a motor vehicle with four wheels; usually propelled by an internal combustion engine'
>>> for synset in wn.synsets('car')[1:3]:
print synset.lemma_names
['car', 'railcar', 'railway_car', 'railroad_car']  ['car', 'gondola']
>>> wn.synset('walk.v.01').entailments()# Walking involves stepping
[Synset('step.v.01')]
```

Working with NLTK

Regular Expressions

.	Wildcard, matches any character
^abc	Some pattern abc at the start of a string
abc$	Some pattern abc at the end of a string
[abc]	One of a set of characters
[A-Z0-9]	One of a range of characters
ed|ing|s	One of the speci ed strings (disjunction)
*	Zero or more of previous item, e.g. a*, [a-z]*
+	One or more of previous item, e.g. a+, [a-z]+
{n}	Exactly n repeats where n is a non-negative integer
a(b|c)+	Parentheses that indicate the scope of the operators

Examples

Simple paragraph

Thank you all so very much. Thank you to the Academy. Thank you to all of you in this room. I have to congratulate the other incredible nominees this year. The Revenant was the product of the tireless efforts of an unbelievable cast and crew.

Tokenization

1. Thank you all so very much.
2. Thank you to the Academy.
3. Thank you to all of you in this room.
4. I have to congratulate the other incredible nominees this year.
5. *The Revenant* was the product of the tireless efforts of an unbelievable cast and crew.

Pre-Processing

1. thank you all so very much
2. thank you to the academy
3. thank you to all of you in this room
4. i have to congratulate the other incredible nominees this year
5. *the revenant* was the product of the tireless efforts of an unbelievable cast and crew

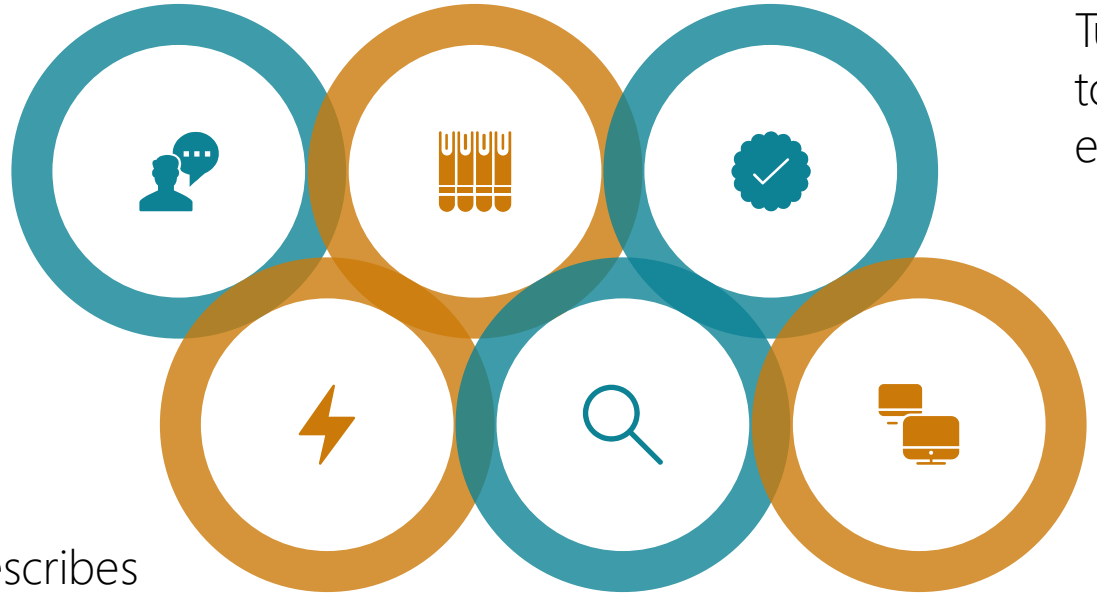
After pre-processing the data will be sent to the actual algorithm.

Conclusion

NLTK is a broad-coverage natural language toolkit that provides a simple, extensible, uniform framework for assignments, demonstrations and projects

It is thoroughly documented, easy to learn, and simple to use. NLTK is now widely used in research and teaching.

Three different types of documentation are available. Tutorials explain how to use the toolkit, with detailed worked examples



The API documentation describes every module, interface, class, method, function, and variable in the toolkit

Technical reports explain and justify the toolkit's design and implementation.

NLTK is available from nltk.sf.net, and is packaged for easy installation under Unix, Mac OS X and Windows.

References

[1] **Natural Language Processing using NLTK and WordNet**, Alabhya Farkiya et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (6) , 2015, 5465-5469 www.ijcsit.com

[2] **NLTK: The Natural Language Toolkit(2011)**, Steven Bird , University of Melbourne, Edward Loper, University of Pennsylvania sb@csse.unimelb.edu.au http://arXiv.org/abs/ cs/0205028.

[3] **Using NLTK for educational and scientific purposes**, Mykhailo Lobur, Andriy Romanyuk, Mariana Romanyshyn, CADSM'2011, 23-25 February, 2011, Polyana-Svalyava (Zakarpattya), UKRAINE

[4] **NLTK: The Natural Language Toolkit,** Edward Loper and Steven Bird Department of Computer and Information Science University of Pennsylvania, Philadelphia, PA 19104-6389, USA

[5] **NLTK BOOK**: https://www.nltk.org/book/

Thank You