**Implementation**

**4.1 System API**

The API is developed in Python language which is platform independent i.e. it works the same on all the operating systems irrespective of their build as the libraries required for Python are usually based on C/C++. So as soon as these libraries are loaded, it is good to go.

The module has to be run by the user in order to load all the sub modules of the software package. As user sends the command to load main module named ‘TextA.py’, the module is run. This module loads following sub modules and waits for user for further inputs.

* **TextExtractor**
* **TextLemmatiser**
* **TextNLGenerator**
* **TextREChunker**
* **TextTagger**
* **TextTokenizer**

**4.2 User interactions**

The user is supposed to run the main module ’TextA.py’ in the python console on any platform. The module loads all other necessary sub modules.

The system then asks the user to provide a paragraph or a sentence for parsing and processing through Morph dictionary.

System runs the input through all modules and provides the user with various outputs based on the module and Morph dictionary used for processing.

**Given is a snapshot of the software in run:**

**4.3 Main module (TextA.py) description**

**Following is the description of various classes used in main module to perform various tasks:**

1. extract\_info(self,chunked\_text):

* extracts many useful things from chunked\_text
* outputted in a dictionary

1. lemmatise\_tagged(self,tagged\_text):

* lemmatizes tagged text and outputs the form:

'These/DT/These sentences/NNS/sentence were/VBZ/be false/JJ/false'

1. chunk\_lemmatised(self,lemmatised\_text):

* inputs lemmatized text of the form:

"He/PRP/he ran/VBD/run"

* outputs the form:

"(NX He/PRP/he NX) (VX is/VB/be VX) (NX the/DT/the mailman/NN/mailman NX)"

1. chunk\_tagged(self,tagged\_text):

* chunks tagged text and outputs the form:

"(NX He/PRP NX) (VX is/VB VX) (NX the/DT mailman/NN NX)"

1. parse\_pred\_arg(self,pp):

* parses the predicate-argument string
* returned by jist\_predicates(), of the form:

'("pred name" "arg 1" "arg 2" etc)'

* and returns them as a list

1. strip\_tags(self,tagged\_or\_chunked\_text):

* strips part-of-speech and chunk tags from text
* and returns plaintext

1. tag\_tokenized(self,tokenized\_text):

* takes tokenized text and returns Penn Treebank tagset tagged text:

i.e.: "This/DT is/VB a/DT sentence/NN".

1. tokenize(self,sentence,expand\_contractions\_p=1):

* inputs a raw text sentence and outputs that sentence with punctuation tokenized, except in the case of abbreviations:

(e.g. "can't"-->"cannot")

1. split\_sentences(self,text):

* input a raw text and outputs a list of sentence segments

1. split\_paragraphs(self,text):

* inputs a raw text and outputs a list of paragraph segments

1. pp\_info(self,extracted\_infos):

* pretty prints sentence information digests returned by jist()

1. jist(self,text):

* inputs raw text, outputs a list of dictionaries with information digests of each sentence

1. jist\_predicates(self,text):

* similar to jist() except output is simpler
* returns a list (document-level) of lists (sentence-level) of lisp-style predicate argument structures
* each structure should look something like this:

("verb" "subject" "obj1" "obj2" ...)

1. generate\_sentence(self,vsoo,sentence\_type='declaration',tense='past',s\_dtnum=('',1),o1\_dtnum=('',1),o2\_dtnum=('',1),o3\_dtnum=('',1)):

* inputs verb-subject-object-object tuple
* outputs a generated sentence
* valid sentence types: declarative, imperative, (can|may|would|should|could), (who|what|when|where|why|how), question
* valid tenses: past, present, progressive, past\_progressive, future, infinitive

1. generate\_summary(self,vsoos):

* uses TextNLGenerator to generate a paragraph text summary in the past tense
* inputs a flat list of verb-subject-object-object tuples

**4.4 Sub module description**

1. **TextTokenizer-**

* It normalizes punctuation, spacing and contractions, with sensitivity to abbreviations.

1. **TextTagger-**

* Part-of-speech tagging using PENN TREEBANK tag-set
* enriched with "Common Sense" from the Open Mind Common Sense project
* exceeds accuracy of Brill94 tagger using default training files

1. **TextREChunker-**

* chunks tagged text into verb, noun, and adjective
* chunks (VX,NX, and AX respectively)
* incredible speed and accuracy improvement over previous TextChunker

1. **TextExtractor-**

* extracts verb-argument structures, phrases, and other semantically valuable information from sentences and returns sentences as "digests"

1. **TextLemmatiser-**

* part-of-speech sensitive lemmatization
* strips plurals (geese-->goose) and tense (were-->be, had-->have)
* includes regular expressions from Humphreys and Carroll's morph.lex, and PENN's XTAG corpus

1. **TextNLGenerator-**

* generates summaries
* generates surface form sentences
* determines and numbers NPs and tenses verbs
* accounts for sentence type