**CHAPTER 7**

**SYSTEM DESIGN**

**7.1 Block Diagram**

The system is divided into the following subsystems:

****

***Fig 7.1*** *Block diagram of the system*

7.1.1 Python Back-end

The software will analyse the text input by the user and analyze it for different

rules and styles. This will be in turn be using NLTK framework and the data

available in various corpora and dictionaries, etc.

7.1.2 User Subsystem

The user subsystem will represent the users of this system. The user will interact with our system by creating/opening a .txt file in the editor window.

7.1.3 NLTK

All language processing will be done using NLTK framework. It is an open

source, free toolkit that is written in python.

7.1.4 NLTK data

The NLTK framework comes with support for various corpora, dictionaries, etc.

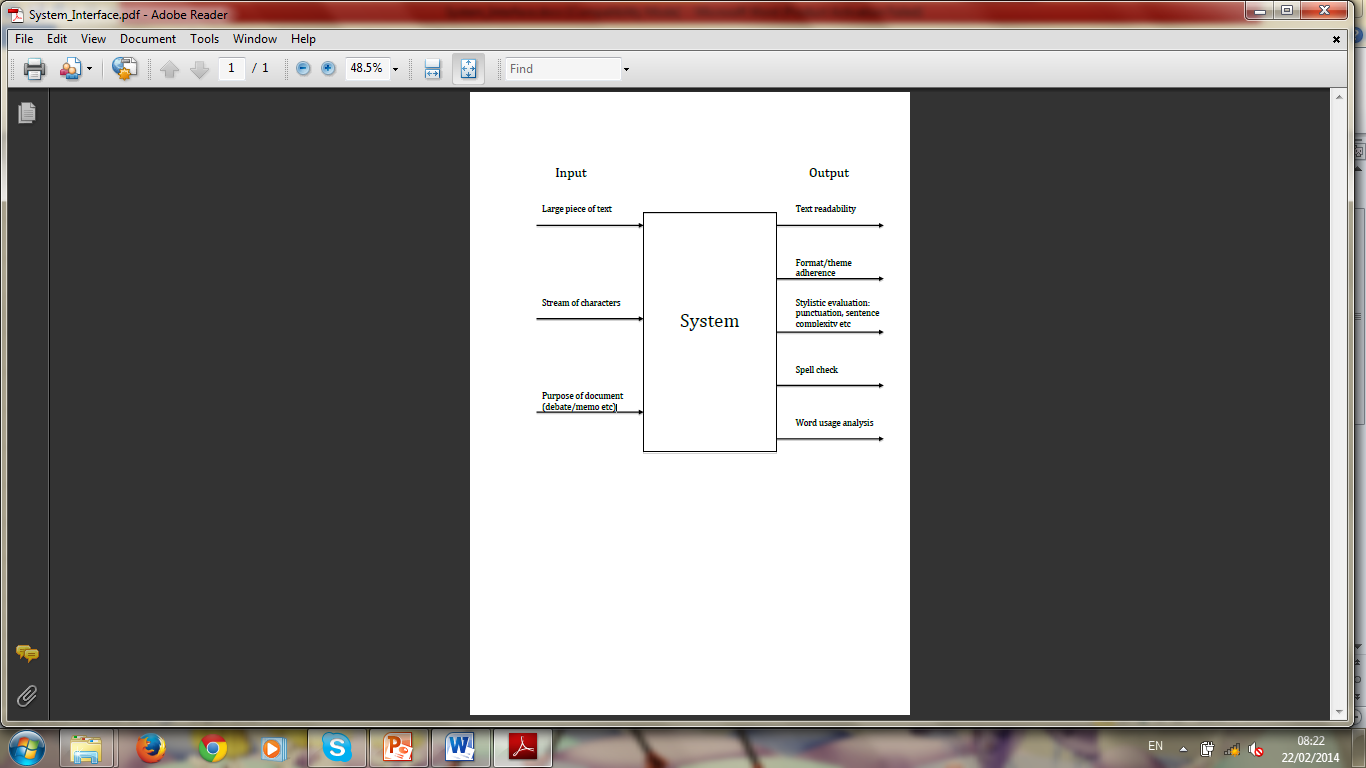
The corpora used are Brown and Wordnet. CMU dictionary is also used as a reference for some functionality.

7.1.5 PyQT User Interface

The user interacts with the system by the means of an editor window. This subsystem is responsible for the flow of data to the back end and display of comments, etc.

**7.2 Architecture**

As mentioned earlier the user interacts with the system through an editor window. The user can choose to either type directly into the window or open an existing file. The user can also choose to specify the purpose of the document. The output generated is the readability of the text, template for the chosen format, stylistic evaluation, word usage analysis, floweriness, obscurity, etc.



***Fig 7.2*** *Architecture for the proposed system*

The major phases in the processing of the text input by the user are as follows:

* Creating and saving a text file
* Giving the user a template for the formats
* Sentence level analysis
* Word level analysis
* Calculating readability of the text
* Calculating floweriness and obscurity of the text
* Generating comments in the editor window

**7.2.1 Creating and saving a text file**

The text to be processed can be directly typed into the editor window. If the user wishes to save then in his system then the new file can be saved into a directory with a given name. The user can also open an existing file. The data is loaded into the editor window.

**7.2.2 Giving the user a template for the formats**

The following templates are provided to the user for consideration. These templates open a dialogue box when chosen. There are various text boxes in the dialogue box, one for each section of that particular format.

7.2.2.1 Formal Letter: A formal letter should have the following:

* Sender’s address
* Receiver’s Name
* Receiver’s Address
* Date
* Subject
* Salutation
* Content
* Regards
* Sender’s Name

7.2.2.1 Debate: A debate should have the following:

* Introduction
* Argument 1 followed by justification
* Argument 2 followed by justification
* Argument 3 followed by justification
* Conclusion

7.2.2.3 School Essay: A good school essay should have the following:

* Title of the essay
* Introduction of the topic
* Main point #1 followed by its description
* Main point #1 followed by its description
* Main point #1 followed by its description
* Conclusion

**7.2.3 Sentence level processing**

The following is done as a part of sentence level analysis of the text:

7.2.3.1 Sentence tokenization: The text is broken into sentences for them to the analysed. This is done using the functions provided by NLTK.

7.2.3.2 Check if sentence is too long: The length of the sentence is compared with a hard-coded value that generates comments about the sentence being too long if true.

7.2.3.3 Check if the sentence is in passive voice. If the use of passive voice is detected in a sentence, appropriate comments are generated in the editor window.

7.2.3.4 Check if any commonly misused phrases are present in the sentence: A hard-coded list of misused phrases is looked for in the text. If any of them are present, corrections are suggested.

**7.2.4 Word level processing**

The word level processing of the text involves the following:

7.2.4.1 Tokenizing each sentence into words: Word tokenizing is done using functions provided by the NLTK.

7.2.4.2 Tagging of words: The tokens (words) are tagged as parts of speech based on their context in the text. The POS tagging feature of NLTK is used for this purpose.

7.2.4.3 Check if the last word in the sentence is a preposition: Based on the tag of the last word in the sentence, it is judged whether the sentence ends with a preposition or not

7.2.4.4 Check if the words are misused: The tagged words are compared against some hardcoded values. If the tag doesn’t match the hard-coded value, alternate words are suggested.

**7.2.5 Calculating readability of the text**

The following reading scores are generated:

7.2.5.1: Flesch Reading Ease:

206.835 - 1.015 \* (total words / total sentences) - 84.6 \* (total syllables / total words)

The higher the score, the easier it is to read.

7.2.5.2 Flesch-Kincaid Grade Level:

0.39 (total words / total sentences) + 11.8 (total syllables / total words) - 15.59

Lowest theoretical grade score: -3.4

7.2.5.3 Gunning-Fog Score:

0.4 \* [(words / sentences) + 100 \* (complex words / words) ]

7.2.5.4 Coleman-Liau Index:

0.0588L - 0.296S - 15.8

L: average number of letters per 100 words

S: average number of sentences per 100 words.

7.2.5.5 SMOG Index:

1.043 \* √ (number of polysyllables \* 30 / number of sentences) + 3.1291

Polysyllables: words with 3 or more syllables

7.2.5.6 Automated Readability Index:

4.71 \* (characters / words) + 0.5 \* (words / sentences) - 21.43

**7.2.6 Calculating floweriness and obscurity of the text**

7.2.6.1 Floweriness: The floweriness of the text is determined by the number of adjectives and adverbs per sentence in the text. The tag of each word in the sentence is checked. All the words tagged adjectives and adverbs are counted. This number is divided by the total number of sentences in order to generate the floweriness.

7.2.6.2 Obscurity: The frequency (its usage in English language) of each word in the sentence is found. The words with frequency less than 5 are considered rare/difficult. All such words are counted and the total divided by the total number of sentences in the text to generate the obscurity.

**7.2.7 Displaying comments in the editor window:**

All the comments generated by the back-end subsystem need to be displayed in the main editor window. The readability scores are calculated either automatically (triggered by the user by pressing the space bar) or manually, when the user presses either Ctrl+G or clicks on the calculate readability score under the tools section.

The evaluation of the text is done when the user presses Ctrl+E or clicks on Evaluate text option under the tools menu.

**7.3 Use cases:**

3.4.1 Creating a document

3.4.1.1 Actor: Generic user

3.4.1.2 Description: The user can use the application to create a document.

Once the file has been created, he/she can type text and save the document.

3.4.1.3 Pre-condition: Unopened application.

3.4.1.4 Post-condition: Blank canvas for the user to start typing text.

3.4.2 Evaluating stylistically

3.4.2.1 Actor: Generic user

3.4.2.2 Description: The user can use the application to obtain comments on his style of writing. The application detects passive voice usage, inconsistent tenses, provides information on word count, etc.

3.4.2.3 Pre-condition: A substantial amount of text written by the user.

3.4.2.4 Post-condition: Comments against the corresponding line number.

3.4.3 Checking word usage and word frequency

3.4.3.1 Actor: Generic user

3.4.3.2 Description: The user can use the application to obtain data on how common the words he has written in the document are and words that he has used too often.

3.4.3.3 Pre-condition: A substantial amount of text written by the user.

3.4.3.4 Post-condition: Comments on words that have been used very frequently. A rating of the complexity of the text is based on universal word frequency.

3.4.4 Conforming to a template

3.4.4.1 Actor: Student

3.4.4.2 Description: A student can use the application to adhere to a particular template while writing a document. This is a preference given prior to document editing and hints are provided as the student prepares the document.

3.4.4.3 Pre-condition A blank document

3.4.4.4 Post-condition Hints to assist the student to adhere to the template (ex: debate)

3.4.5 Analysing readability

3.4.5.1 Actor: Novelist

3.4.5.2 Description: The novelist or writer can use the application to obtain an analysis on the readability of the text written. The readability scores are calculated from universally accepted formulae.

3.4.5.3 Pre-condition: A complete document

3.4.5.4 Post-condition: Readability scores on the document.