ALGORITHM FOR TEXT TO GRAPH CONVERSION AND SUMMARIZING USING NLP: A NEW APPROACH FOR BUSINESS SOLUTIONS

Prajakta Yerpude and Rashmi Jakhotiya and Manoj Chandak
Department of Computer Science and Engineering, RCOEM, Nagpur

Abstract

Text can be analysed by splitting the text and extracting the keywords .These may be represented as summaries, tabular representation, graphical forms, and images. In order to provide a solution to large amount of information present in textual format led to a research of extracting the text and transforming the unstructured form to a structured format. The paper presents the importance of Natural Language Processing (NLP) and its two interesting applications in Python Language: 1. Automatic text summarization [Domain: Newspaper Articles] 2. Text to Graph Conversion [Domain: Stock news]. The main challenge in NLP is natural language understanding i.e. deriving meaning from human or natural language input which is done using regular expressions, artificial intelligence and database concepts. Automatic Summarization tool converts the newspaper articles into summary on the basis of frequency of words in the text. Text to Graph Converter takes in the input as stock article, tokenize them on various index (points and percent) and time and then tokens are mapped to graph. This paper proposes a business solution for users for effective time management.

Keywords

NLP, Automatic Summarizer, Text to Graph Converter, Data Visualization, Regular Expression, Artificial Intelligence

1. Introduction

The paper deals with applications of natural language processing using its various domains regarding textual analysis. Natural language processing (NLP)[1] is a bridge between human interpretations and computer. It makes use of artificial intelligence and various techniques of analysis to give about 90% accuracy of data. The term Natural Language Processing [4] comprises a great horizon of techniques for automatic generation, manipulation and analysis of natural or human languages. It includes various categories like syntactic analysis[22]where sequence of words are converted to structures that shows relation between the words, semantic analysis[9] where meanings are assigned to a group of words, pragmatic analysis[24] where differences between expected and actual interpretation is analysed, morphological analysis[10] where punctuations are grouped and removed etc. The paper demonstrates two different types of applications that use NLP principle and are as follows:

- An automatic text summarizer

DOI: 10.5121/ijnlc.2015.4403

Domain: Newspaper articles

- Statistical unstructured text to graph conversion

Domain: Stock market articles

The above applications deal with textual analysis and deriving an optimum result to reduce the time of any reader. Often it becomes tedious for any reader to read and interpret the whole article from any newspaper whether it belongs to any domain. Hence it becomes necessary to optimize this data by removing redundancies in an efficient way. Natural Language Processing provides various techniques for text processing and is available in various technologies like Python, Java, Ruby, etc. The technology used for these two applications is Python which provides with NLTK- Natural Language Toolkit [4] that provides various types of libraries for textual analysis. Python provides with extensive approach to the Regular Expressions and NLP required for text processing.

Automatic summarization deals with removal of redundancy from the text thereby maintaining the gist of any text. There are techniques available for textual analysis which includes text processing, text categorization [13], part of speech tagging [20], and regular expressions [8] to classify text and summarize it. Methods of summarization include extraction [20], where main keywords and sentences are returned as a summary whereas abstraction refers to building of a new text based upon the content. The paper focuses on extraction method that provides insight to text analysis. There are API's of summarization available in Java that consumes memory as well as time for processing. Python, being equipped with NLTK [15] provides an efficient way for implementing NLP tasks, thereby reducing time and space of the user. We have used Python for implementing summarizer.

Statistical data includes figures, comparison of two different datasets, numbers that are easily understood when explained using visual aid. Graphs are used as a visual aid for representation statistical data in an efficient way. There are tools available that convert structured data to graphs like Microsoft Excel where figures have to be entered manually which becomes quite tedious. Python consists of libraries for plotting graphs from given lists of tokens of texts. Our focus is to convert unstructured data into a graphical format by extracting figures [4] and arranging them in a data structure named 'dict' in Python [14].

Software Development Lifecycle [18] gives a systematic approach to the development of any software. The phases of module implementation were planned, designed, coded, tested and integrated. Planning included requirements gathering, technological study, survey of text and deciding upon flow of working. Designing and Coding included the implementation of stepwise approach to the tool. Testing included construction and implementation of various use cases to determine the viability of tool.
The organization of the latter part of the paper is as: chapter 2 gives the background and related work done in the area of NLP and its applications using various technologies and the advantages of the technology used in the project is explained. Chapter 3 gives our components details of the

Python and NLTK, which we adopted for implementation as a part of our project on NLP. Chapter 4 shows the experimental details of both the applications and the flow of working of the programs. Chapter 5 summarizes the whole paper with conclusion and describes the future scope in the field of NLP.

2. Related Work:

NLP is an important area of research in many direct or indirect application problems of information extraction, machine translation, text correction, text identification, parsing, sentiment analysis, etc. Our work has the major focus on information extraction i.e. getting the important words, figures from the text.

The two projects Automatic text Summarizer and Text to Graph Conversion both require extraction of text. In the former, the text is entered and the tokens are extracted to calculate frequency which on integration would return the sentences according to the highest rank obtained helping in creation of summary. While in the later, tokens are again extracted in the form of points, percentage, time, company, etc which are stored in data structure known as dictionary and mapped onto the graph.

The technology used is Python. Python consists of 'n' number of libraries for simplified processing of textual data. Python is used to handle various tasks of NLP which include parts of speech tagging, classification, translation, noun phrase extraction, etc. Researchers of NLP and programmers have developed multiples ways of text summarization and various online tools using extractive techniques.

Most early focus of automatic text summarization was on technical documents. The most cited paper on summarization is that of [11], describing the research done at IBM in the 1950s. Related work [2], also done at IBM, providing early insight on a particular feature assisted in finding important parts of documents: the sentence position. Some research processes [7] describe a system that produces document extracts. His primary contribution was to develop a typical structure for an extractive summarization experiment.

Many tools are available wherein the information has to be entered in the structured format and is used to map that information on the graph. In most of the cases, csv (comma separated values) file, excel files or any structured data source is to be attached to the tool in order to get graphical representation of the information present in the document. Various platforms for conversion of structured information to graph are Microstrategy, MS-Word, MS-Excel, Tableau, etc.

Our research focuses on extracting the text from the stock articles which is in unstructured form and then maps them to the graph. Our research has an additional feature of extracting tokens from the unstructured document which is based on text processing in NLP. The text classification[17] has been a subject of ongoing researches to get the in-depth knowledge of various types of languages and their profound meanings. Some languages like Chinese and Japanese where sentences determine the limit have to undergo word segmentation[5] process that also removes the whitespaces between the words. This approach has been used to remove the white-spaces between words in text.

Various researches and programs have been developed using Java as technology. But for text processing, Python has few added advantages over Java. Python has various libraries for text processing like NLTK (Natural Language Toolkit) [15], TextBlob [24], Pattern [16], etc. Python is less verbose as compared to Java. It requires about 10 lines of code for a program in Java,

while it requires only 2 lines of code in Python. As it is dynamically-typed language, it is estimated that programmers in Python can be 5-10 times productive than that in Java, which is statically-typed. The input text can be taken from web pages using BeautifulSoup[3]. Python[23] has extensive standard libraries which bolster everything from string and regular expression processing to XML parsing and generation.

2.1 Text Segmentation:

Segmentation[5] involves splitting of text into key phrases, words and tokens. Like Google shows the most relevant results during the search, Text segmentation gives this result by Information Retrieval[25]. This process include approaches like stopword removal, suffix stripping and term weighing to calculate the most important keyword of the text. Stopwords are those words that cause redundancy in the text. Words like a, an, the, to, in etc. are considered as stopwords. The terms are weighed according to their frequencies in text. Certain algorithms like TextTiling[25] break up the text into multiple paragraphs(subparts) by semantic analysis. In this paper, the text mapping is done using regular expressions for deriving patterns and information retrieval techniques like stopword removal and term weighing are used.

3. Operations Used For Text Segmentation:

3.1 Components for text analysis:

1. Collections: Collections contains different types of modules out of which defaultdict(x) is used to declare and define a variable of any data type 'x'.This data structure uses of keys and their corresponding values as a pair and stores them accordingly. Associative arrays and hash tables also make use of python dictionaries where functions are mapped with their pointer values as addresses. The general syntax of a dictionary is given below:

dict = {p(key):x(value)}

Example: *dict1 = {'9:00 am': '27,890.09', '4:00 pm': '26,990.01'}*

2. Heapq: This python module gives a structured and systematic implementation of heap queue algorithm. In heapq, given a particular list can be converted to a heap by means of the heapify() function. The method nlargest() was used to get the most important ‘n’ sentences.[This module is used in text summarizer in order to fetch ‘n’ sentences as required by the user in summary]

3. Nltk.tokenize: Tokens are the substrings of a whole text. Hence tokenize method is used for splitting any string into substrings according to the conditions provided. From this module the methods sent_tokenize[4] and word_tokenize were used. Sent_tokenize splits the input text (paragraph) into sentences while word_tokenize divides these sentences into words.

If a sentence is 'History gives information about our ancestors'

>>word_sent = [word_tokenize(s.lower()) for s in sents]

>> print word_sent

[['History', 'gives', 'information', 'about', 'our', 'ancestors', '.']

4. Nltk.corpus: Corpora contain a large set of structured data. In Python, a collection of corpus contains various classes which can be used to access these large set of data. Stopwords

are most common words such as the, is, on, at, etc. The method call stopwords.words('text') was used to remove these unimportant words.

For example:

>>> from nltk.corpus import stopwords

>>> a= set(stopwords.words('english') + list(punctuation))

>>> print a

set([u'all', u'just', u'being', '-', u'over', u'both', u'through', u'yourselves', u'its', u'before', '$', u'herself', u'had', ',', u'should', u'to', u'only', u'under', u'ours', u'has', '<', u'do', u'them', u'his', u'very', u'they', u'not', u'during', u'now', u'him', u'nor', '`', u'did', '^']) and so on.

3.2 Components for Pattern Matching:

Regular expressions re: Regular expressions[14] abates the time for processing the whole text by providing various simple and easy to use formats for text searching patterns[21], replacing and their analysis[8].

1. re.**search**(*pattern, string*)

This method scans the text and checks the location where the pattern matches and regular expression returns the matching object's instance. It returns nothing if the pattern is not matched in the string.

Example: *>>>p = re.search('(?<=abc)points', 'mainpoints')*

>>>p.group(0)

>>>'points'

2. re.**match**(*pattern, string*)

This method matches zero or more characters of the pattern at the beginning of a string and if matched, returns its corresponding matching object instance. Similarly like search method, it returns nothing if the pattern is not matched in the string.

Example: *>>> u = re.match(r'(\w+) (\w+)', 'Lord Tyrell, King')*

>>>u.group(0)

>>>'Lord Tyrell'

3. re.**findall**(*pattern, string*)

This method returns the matches of pattern in the form of list of strings. While scanning the text sequentially, it returns the found matches in order. It also returns any matched group in the form of a list. If matches are not found, empty lists are included in the group. A list of tuples will be returned if the pattern contains different groups.

4. re.**compile**

This method is used to execute a regular expression. The conditions specified in the expressions are checked and results are returned.

5. re.**strip**

It is applied on a string or a string of characters to remove or hide the invalid elements. It is also used in bifurcation of the text according to the conditions specified to reduce time for scanning.

3.3 Components for Database Connectivity:

1. MySQLdb:

MySQLdb is a compatible interface to MySQL database server that connects database in MySQL. The next step to using MySQL in a Python script is to make a connection to the database [12].All Python Database-API 2.0 modules provide a function 'database_name.connect'. This is the function that is used to connect to the database, in our case MySQL.

>>>db(anyname)=MySQLdb.connect(host=HOST_NAME,user=USER_NAME,passwd=MYPASSWORD, db=DB_NAME)

In order to put our new connection to good use we need to create a cursor object. The cursor object is used to work with the tables of database specified in the Python Database-API 2.0. It gives us the ability to have multiple separate working environments through the same connection to the database. One can create a cursor by executing the 'cursor' function of your database object.

>>>cur(name) = db.cursor()

Executing queries is done by using execute() method.

3.4 Components for Graphical User Interface:

Tkinter: Tkinter[14] is the Python module for implementing GUI programming where it provides functions like buttons to navigate, message and dialogue boxes for entering text, scrollbars, text widgets and design templates for GUI. Text widget is where multiple lines can be written in a text box and Tkinter provides flexibility for working with widgets. They are also used for showcasing web links and images. Distributions of TK module are available for Unix as well as Windows.

Example: To create a new widget,

>>>import Tkinter

>>>new = Tkinter.Tk()

>>>new.mainloop()

PIL module is used for inserting graphics such as images and videos on GUI. Images in formats like BMP, JPEG, CUR, DCX, EPS, FITS, FPX, GIF, etc are supported by this library.

3.5 Components for Graph Plotting:

Matplotlib: Python provides a 2D plotting library for line graphs, bar graphs, pie charts, histograms, scatterplots etc. Matplotlib can be used in Python shell as well as script, html servers and GUI toolkits[26]. Simple plotting can be combined with iPython provides a Matlab type interface. This module lets you deal with the object oriented concepts thereby letting user a full control over its working. PlotPy is imported for Matplotlib which provides collection of various styles for plotting. Functions like constructing a graph, changing variables, plotting area and lines in that area, labels can be easily implemented in PlotPy. A bar and line graph is used to store statistical information about stock articles in this project.

Example: To plot a line graph,

>>>plt.plot([1,2,3,4],[1.5,2.5,3.5,4.5]) >>>plt.ylabel('numbers') >>>plt.xlabel('decimal numbers') >>>plt.show()

Functions of Modules used in Project:

Table 1: Modules of Python

Modules	Classes	Functions
Text Analysis		
NLTK-Natural Language Toolkit	Tokenize Corpus	Word.tokenize() Sentence.tokenize() Stopwords()
Collections	Dict	Defaultdict()
RE-Regular Expressions	Re	Replace() Split() Compile() Strip() Findall()
String	Punctuation	Append()
Graph Plotting		
MatPlotlib	Pyplot	Figure() Plot() Bar() Line()
Database Connectivity		
MySQLdb	Mdb	Connect() Cursor() Fetchall()

4. Experimental Details:

4.1 AUTOMATIC TEXT SUMMARIZATION USING NLTK IN PYTHON

A summary states the most important points of the text in a shorter form. It helps to retain the gist without having to go through irrelevant information also the reader can decide if going through the entire document is actually necessary or not. A text summary restates the important points of text in a compressed form. It presents only salient information, in a condensed format. Thus it helps the reader to get acquainted with the subject matter and also to decide whether reading the entire document will be useful. Automatic Text Summarization has two general approaches: extraction and abstraction. Abstraction works in a way similar to the way humans

would summarize text. It first builds an internal semantic representation and then generates a summary with the help of Natural Language Generation Techniques. The extractive method selects from the original text a subset of words, phrases, sentences. These are then arranged in proper sequence to give the summary. This summarizer was developed using the extractive technique wherein the most important sentences are extracted and retained in the summary.

Flow of Working of Algorithm:

Input: News article Output: Summarized article

Steps: [In brief]

1. The news article and the no. of statements required in the summary are entered.
2. The entered text is split into sentences. These sentences are then split into words.
3. These words are filtered by removing stopwords and punctuations.
4. The frequency of each word (from remaining words) is calculated.
5. The frequency of each word relative to the word having highest frequency is calculated.
6. The rank of each statement in the input text is calculated by adding up the relative frequencies of the words appearing in those statements.
7. These sentences are then sorted using nlargest method of heapq which returns 'n' sentences having highest ranks.
8. These sentences are then returned as summary statements.

Detailed Explanation:

Take the text as an input and tokenize it into sentences and words using ntlk.tokenize modules namely sent_tokenize and word_tokenize and filtering the words by removing the stopwords using nltk.corpus module. On sent_tokenize the entered text gets split on a period (.) and sentences are obtained. These sentences are further operated on by word_tokenize to obtain tokens in the form of words. Stopwords are the words likes articles, to be verbs (am, is are, was, were, etc) and also the punctuations of which if the frequency is calculated will just increase the complexity of the code. So these words are to be neglected as soon as the text is entered.

```
sents = sent_tokenize(text)              // 'sents' contains sentences
word_sent = [word_tokenize(sent)]   // 'word_sent' contains words
a= set(stopwords.words('english') + list(punctuation))      // 'a' contains stopwords
```

The next step is calculating the frequency of words which belong to 'word_sent' and not to set 'a' containing stopwords. The frequency calculated is stored using 'collections' module in Python.

```
freq[word] += 1
```

The further part is calculating the rank of each sentence. The frequency of each word in a sentence is integrated and a rank is given to each sentence and the sentences are sorted in descending order of the rank. This is done using sort method using heapq module in Python Language.

```
rank[sent] += self._freq[word]
```

The last part is displaying the summary. So the highest ‘n’ sentences are returned as summary of the entered text. The GUI (Graphical User Interface Is created In Python Using Tkinter module for working of this program).Here the input entered is the text and the number of statements in which summary is required. The output is the summary, number of statements in the entered text and the Summary Ratio (%).

Figure 1 Output of text summarizer

4.2 AUTOMATIC TEXT TO GRAPH CONVERSION USING NLP IN PYTHON

Graph is a good method of condensing and representing data in a readily understandable form. The visual representation provides an ease of access to the statistical data and interpret data at a glance. Graphical representation makes data easy to recall. Our tool focuses at automatic conversion of statistical data that comes with stock market into graphs. The automated graph enables to overview and to explore the statistical data sets and has a great potential to research. Graphs are of immense importance in decision making in business, marketing etc. The domain used here is stock news articles. Text processing is efficiently handled in Python language which itself is integrated with natural language processing. Python consists of huge libraries for graph plotting, database connectivity and textual analysis which proved significant in designing the tool. NLTK is a platform that enables Python programs to work with natural language. It provides a collection of classes, modules, methods, etc. making it easier to process text. Our domain, Stock articles, consists of statistics and figures in different formats like time, index (points and percent). Tokens, figures were extracted using NLTK and regular expressions respectively whereas mapping of graphs is done using Python libraries.

Stock market is where dealers and buyers come across and trading occurs between them, and with it comes a lot of figures in the form of shares. The main aim is whether the statistical

information from the stock news can be directly converted to a graph which will be very easy to access and compare. Our tool have focused the BSE (Bombay Stock Exchange)and the private sector companies to derive the daily gainers and losers. It gives two graphs containing the sensex points and losers and gainers. Stock articles have been extracted from Economic Times[6]which gives the scenario of the entire day.

Flow of working of Algorithm:

Input: Stock Article Output: Graphical Representation

Steps: [In brief]

Note: Establish Database Connectivity

PART A: BSE

1. Tokenize the text into sentences on basis of Full stop(Period).
2. Tokenize each sentence into words.
3. Remove all stopwords from the list of words.
4. Traverse the text linearly and match it with the keywords from database.
5. If keyword found, retrieve its tuple from the database and store in list.
6. Retrieve all the figures from the text.
7. Using Regular Expressions fetch the sentences having time mentioned and from it fetch time and stock points to be stored into a dictionary(time:points).
8. For BSE:

a) Plot a bar graph using dictionary where time and points are present

b) Plot a line graph using lists for the figures which don't have time stored against it.

PART B: COMPANIES (GAINERS AND LOSERS)

1. Fetch the sentences from the text containing gainers and losers.
2. In the sentence, fetch the company names from the text, match it with database and store in list.
3. Use Regular Expressions to fetch gain % or lose %
4. If its gain, store it into dictionary 'dict' [key: value] as [company name: +%]
5. If its loss, store it into dictionary 'dict' [key: value] as [company name: -%]
6. Plot the bar graph as companies versus percent (gain or loss)
7. Exit

Detailed Explanation:

The explanation could be better understood using an example as follows:

Enter the text: The S&P BSE Sensex started on a cautious note on Wednesday following muted trend seen in other Asian markets. The index was trading in a narrow range, weighed down by losses in ITC, ICICI Bank, HDFC Bank and SesaSterlite. Tracking the momentum, the 50-share Nifty index also turned choppy after a positive start, weighed down by losses in banks, metal and FMCG stocks. At 09:20 am, the 30-share index was at 27419, down 6 points or 0.02 per cent. It touched a high of 27460.76 and a low of 27351.27 in trade today. At 02:30 pm, the 30-share index was at 27362, down 62 points or 0.23 per cent. It touched a high of 27512.80 and a low of 27203.25 in trade today. BHEL (up 1.2 per cent), TataMotors (up 1.1 per cent), HUL (up 1.1 per cent), Wipro (up 1.03 per cent) and Infosys (up 0.74 per cent) were among the major

Sensex gainers. ITC (down 2.8 per cent), SesaSterlite (down 2.3 per cent), Hindalco (down 1.3 per cent), ICICI Bank (down 0.94 per cent) and TataSteel (down 0.66 per cent) were the major index losers.

1. After entering the input text, stopwords will be removed using the function
stopwords = list(stopwords.words('english') + list(punctuation))
and main tokens will be stored in a list as follows:
['p', 'bse', 'sensex', 'started', 'cautious', 'note', 'wednesday', 'following', 'muted', 'trend', 'seen', 'asian', 'markets', 'index', 'trading', 'narrow', 'range', 'weighed', 'losses', 'itc', 'icici', 'bank', 'hdfc', 'bank', 'sesa', 'sterlite', 'tracking', 'momentum', '50-share', 'nifty', 'index', 'also', 'turned', 'choppy', 'positive', 'start', 'weighed', 'losses', 'banks', 'metal', 'fmcg', 'stocks', '09:20', '30-share', 'index', '27419', '6', 'points', '0.02', 'per', 'cent', 'touched', 'high', '27460.76', 'low', '27351.27', 'trade', 'today', '02:30', 'pm', '30-share', 'index', '27362', '62', 'points', '0.23', 'per', 'cent', 'touched', 'high', '27512.80', 'low', '27203.25', 'trade', 'today', 'bhel', '1.2', 'per', 'cent', 'tatamotors', '1.1', 'per', 'cent', 'hul', '1.1', 'per', 'cent', 'wipro', '1.03', 'per', 'cent', 'infosys', '0.74', 'per', 'cent', 'among', 'major', 'sensex', 'gainers', 'itc', '2.8', 'per', 'cent', 'sesasterlite', '2.3', 'per', 'cent', 'hindalco', '1.3', 'per', 'cent', 'icici', 'bank', '0.94', 'per', 'cent', 'tatasteel', '0.66', 'per', 'cent', 'major', 'index', 'losers']

2. After entering the input text, keywords from database are matched with those from the article and the intermediate output will be a list as given below.
('high'),('low'),('today'),('trading')

3. In the next step, regular expressions are applied to this list of words without stopwords and figures are extracted. Below is an example of one such regular expression:
>>abc=re.findall("\d+,\d+.\d+[points]",text)*
>> ['27460.76', '27351.27', '27512.80', '27203.25']
>>time=re.findall("\d+:\d+[am/pm/ a.m./ p.m.]+",text)
>> ['09:20 am ', '02:30 pm']

4. These time and points values are then stored in a data structure dict of python. Following is the list and structure of Dictionary for BSE:
['09:20', '27419']['02:30', '27362']

Table 2: Dictionary for BSE

KEY	VALUE
09:00 am	28450.01
11:00 am	28500.98
01:00 pm	28601.89
03:00 pm	28431.78
05:00 pm	27997.01

5. Similar process is applied for the gainers and losers where positive and negative values are assigned to gainers and losers and both lists are maintained. Regular expressions are applied to store these values in two different lists.

Gainers:*[' BHEL (up 1.2 per cent), TataMotors (up 1.1 per cent), HUL (up 1.1 per cent), Wipro (up 1.03 per cent) and Infosys (up 0.74 per cent) were among the major Sensex gainers. ']*
Losers:*['ITC (down 2.8 per cent), SesaSterlite (down 2.3 per cent), Hindalco (down 1.3 per cent), ICICI Bank (down 0.94 per cent) and TataSteel (down 0.66 per cent) were the major index losers. ']*

6. These lists are then merged into a dictionary to get a unified storage for plotting of graphs. Following is the list and structure of Dictionary for Companies:
{'Wipro': 1.03, 'TataSteel': -0.94, 'BHEL': 1.2, 'TataMotors': 1.1, 'Hindalco': -2.3, 'ICICI': -1.3, 'HUL': 1.1, 'Infosys': 0.74, 'SesaSterlite': -2.8}

Table 3: Dictionary For Companies

KEY	VALUE
Tata Motors	+1.78 %
Cipla	-2.87 %
HDFC	-1.89 %
Bharti Airtel	+2.78 %
Sun Pharma	+1.01 %
ONGC	-1.44 %
NTPC	+3.24 %

7. These lists are passed to PlotPy and Matplotlib modules to generate their respective graphs.

```
figure = plt.figure(figsize = (12,6), facecolor = "white")
ax.set_xlabel('TIME',color='black',fontsize=18)
ax.set_ylabel('STOCK POINTS',color='black',fontsize=18)
ax.set_title('BSE',fontsize=26)
```

xlabel and ylabel is used to give the specifications of what is to be on x and y axis respectively and plt.figure determines the size and colour of the graph.
Figure 2 and Figure 3 gives a screenshot of the output from the given article.

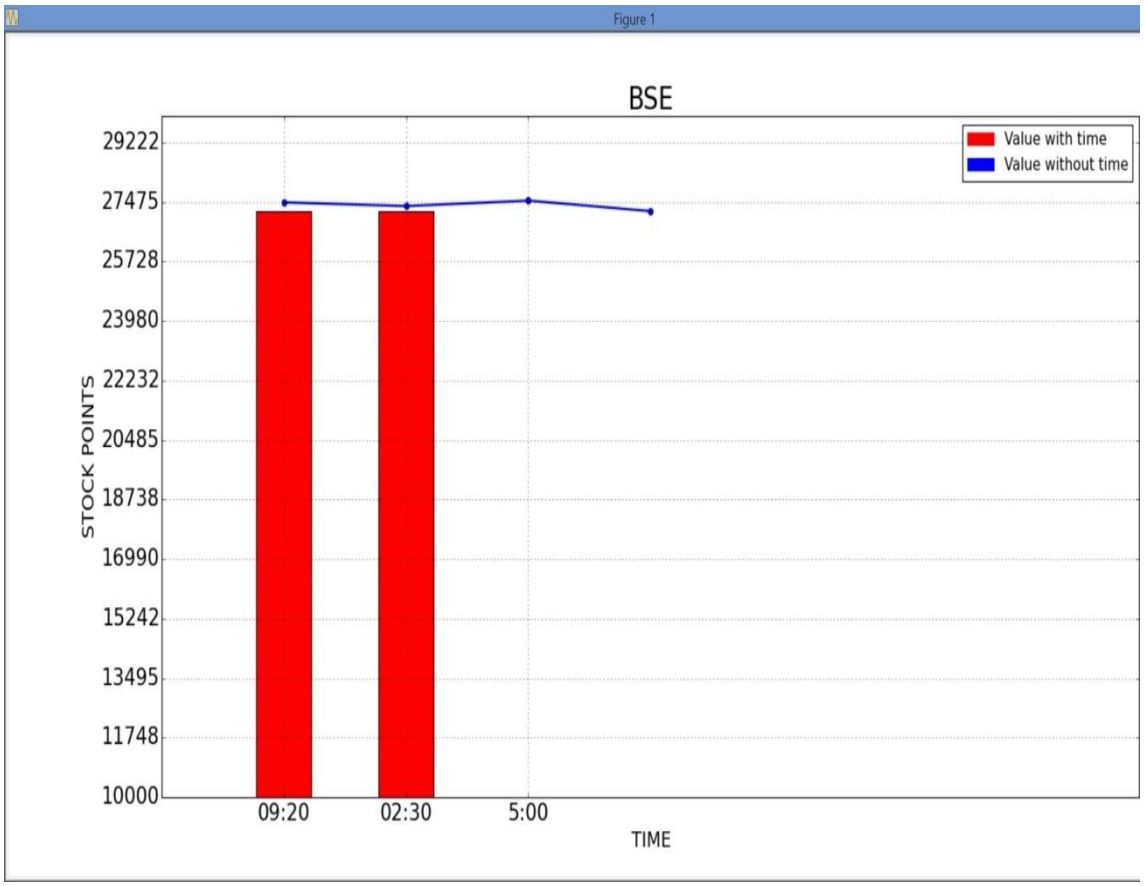

Figure 2 Bse Graph

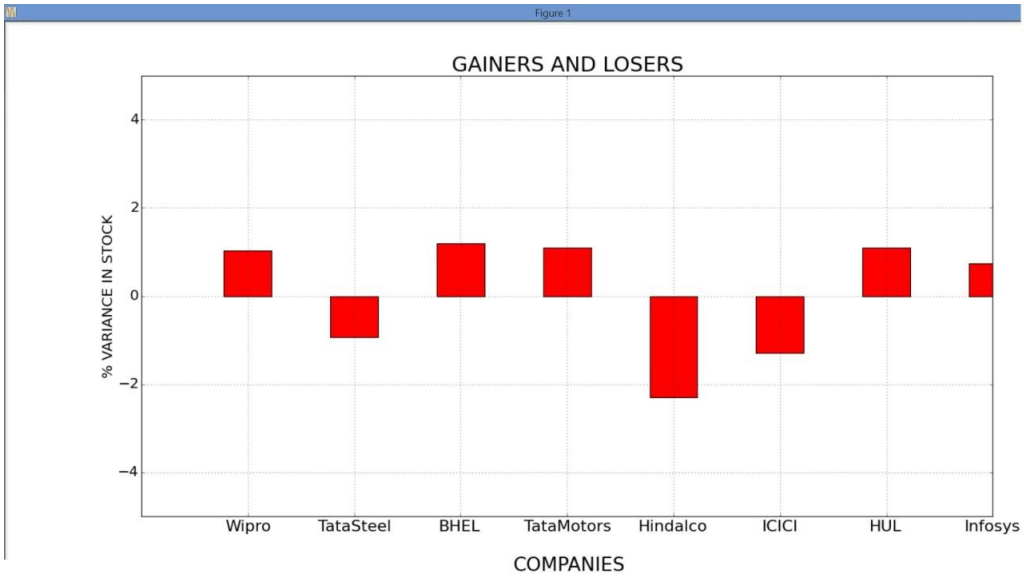

Figure 3 Companies (Gainers And Losers)

5.1 Conclusion:

Due to World Wide Web, the rate of information growth has called for a need to develop efficient techniques to reduce data and make it simpler to understand and convey messages effectively. Natural Language Processing is thoroughly being used worldwide for its efficiency

in text processing. It's delicate to analyze human interpretation using various principles related to NLP. This led to the development of Automatic Summarization tools and Text to Graph Converters. These two applications were focused on text analysis and thereby reducing time to get the main gist of the input processing output.

- Text Summarizer which uses the principle of extraction helps to maintain the pith of what has to be read from a superfluous article, while statistical text to graph conversion eases the method of comparison by directly giving an output as a graph which contains all the required figures.
- In text summarizer, text is analyzed using the frequency of words and ranking a sentence resulting in proper position whereas in text to graph convertor, analysis is done using regular expressions which lets you play with text according to the conditions you specify.
- NLTK, being inbuilt in Python provides a striking approach to information extraction and processing. Thus the text underwent splitting, tokenizing and merging using various modules of NLTK.

5.2 Future Scope

The topics for future scope can be as follows:

- Today, most of the approaches for summarizing are based on extraction. Hence the use of abstraction in text processing which includes automated text building in response to the input can be further implemented.
- Various domains for research in wide spectrum of text to gain accuracy using Python are helpful using frameworks like Django and Flask.
- Text to graph convertors can be used for comparison of datasets having different time domains.
- For handy purposes, these applications can also be converted into a mobile application in Android or iOS.
- Databases can be used for handling multiple datasets and thereby reducing the memory as well as time for retrieval.

References

[1] Allen, James, "Natural Language Understanding", Second edition (Redwood City: Benjamin/Cummings, 1995).

[2] Baxendale, P. (1958). Machine-made index for technical literature - an experiment. IBM Journal of Research Development, 2(4):354–361. [2, 3, 5]

[3] BeautifulSoup4 4.3.2, Retrieved from https://pypi.python.org/pypi/beautifulsoup4

[4] Bird Steven, Klein Ewan, Loper Edward June 2009, "Natural Language Processing with Python**",** Pages 16,27,79

[5] Cortez Eli, Altigran S da da Silva 2013, " Unsupervised Information Extraction by Text Segmentation", Ch 3

[6] Economic Times Archives Jan 2014-Dec 2014, Retrieved from http://economictimes.indiatimes.com/

[7] Edmundson, H. P. (1969). New methods in automatic extracting. Journal of the ACM, 16(2):264–285. [2, 3, 4]

[8] Friedl Jeffrey E.F. August 2006,"Mastering Regular Expressions", Ch 1
[9] Goddard Cliff Second edition 2011,"Semantic Analysis: A practical introduction ", Section 1.1-1.5
[10] Kumar Ela, "Artificial Intelligence", Pages 313-315
[11] Luhn, H. P. (1958). The automatic creation of literature abstracts. IBM Journal of Research Development, 2(2):159–165. [2, 3, 6, 8]
[12] Lukaszewski Albert 2010, "MySQL for Python", Ch 1,2,3
[13] Manning Christopher D., Schütze Hinrich Sixth Edition 2003,"Foundations of Statistical Natural Language Processing", Ch 4 Page no. 575
[14] Martelli Alex Second edition July 2006, "Python in a Nutshell", Pages 44,201.
[15] Natural Language Toolkit, Retrieved from http://www.nltk.org
[16] Pattern 2.6, Retrieved from http://www.clips.ua.ac.be/pattern
[17] Prasad Reshma, Mary Priya Sebastian, International Journal on Natural Language Computing (IJNLC) Vol. 3, No.2, April 2014, " A survey on phrase structure learning methods for text classification"
[18] Pressman Rodger S 6th edition," Software Engineering – A Practitioner's Approach "
[19] Python Language, Retrieved from https://www.python.org/
[20] Rodrigues Mário , Teixeira António , "Advanced Applications of Natural Language Processing for Performing ", Ch 1,2,4
[21] Stubblebine Tony, "Regular Expression Pocket Reference**:** Regular Expressions for Perl, Ruby, PHP, Python, C, Java and .NET "
[22] Sobin Nicholas 2011, "Syntactic Analysis: The Basics", Ch 1,2
[23] Swaroop C H, "A Byte of Python: Basics and Syntax of Python", Ch 5,8,9,10
[24] TextBlob: Simplified Text Processing, Retrieved from http://textblob.readthedocs.org/en/dev
[25] Thanos Costantino ,"Research and Advanced Technology for Digital Libraries", Page 338-362
[26] Tosi Sandro November 2009, "Matplotlib for Python Developers", Ch 2,3

Authors

Prajakta R. Yerpude is pursuing her Bachelor of Engineering 2012-16, Computer Science and Engineering from Shri Ramdeobaba College Of Engineering and Management, Nagpur. She has been working on a domain of Natural Language Processing from two years. Her interests include Natural Language Processing, Databases and Artificial Intelligence. Email: yerpudepr@rknec.edu

Rashmi P. Jakhotiya is pursuing her Bachelor of Engineering 2012-16, Computer Science and Engineering from Shri Ramdeobaba College Of Engineering and Management, Nagpur. She has been working on a domain of Regular Expressions and Natural Language Processing from two years. Her interests include Natural Language Processing, Regular Expressions and Artificial Intelligence. Email: jakhotiyarp@rknec.edu

Dr. M. B Chandak has received his PhD degree from RTM-Nagpur University, Nagpur. He is presently working as Professor and Head of Computer Science and Engineering Department at Shri Ramdeobaba College of Engineering and Management, Nagpur. He has total 21 years of academic experience, with

research interest in Natural Language Processing, Advance networking and Big Data Analytics. Email:hodcs@rknec.edu