Bengali Text Summarization using Statistical and Sentence Similarity Approach

A project paper submitted in partial fulfillment of the requirements for the Degree of Bachelor of Science (Engg.) in Information and Communication Technology

Submitted By

Exam Roll: 1209026
Session: 2012 – 13
Department of Information and Communication Technology
FACULTY OF ENGINEERING
COMILLA UNIVERSITY
KOTBARI COMILLA – 3506
BANGLADESH

Date of Submission: 23rd July, 2018

Dedicated

To

My Dearest Parents and Uncles

Without whose unconditional supports and

continuous inspiration

I may not be able to be here

Abstract

World is now moving in faster speed with the blessings of technology. Information is vastly stored in the cloud instead of hard copy documents or compact disk. Hence, to keep information in short and concise way in the cloud, summarization of information could be a greater choice. Doing manual summarization is obviously tedious task and hence data scientists are thinking of an automated process that provides human quality summary. In this project, we work with two algorithms, namely, statistical and sentence similarity approach. The first approach returns the summary based on frequency of word appearances processing the probability theory while the second figures out the similarity of sentences based on python NLTK corpora and WordNet modules. While testing with several inputs, we observe that the sentence similarity approach gives much better result than statistical approach although it needs a slightly much time. Therefore, sentence similarity could be considered as the best approach of automatic text summarization than statistical approach. Besides, in our project, we choose python as a programming language considering its various advantages like having open source NLTK library, Brown Corpus and WordNet database, integration properties etc.

Keywords: NLTK, Borwn Corpus, WordNet, Sentence similarity, Word order vector, Word similarity, Automatic Text Summarizer.

Table of Content

List of Figures

Chapter 1

Chapter 1 : Introduction

With fastest growing internet facility and technological people currently trust internet more than a bank locker or private safe. People are using internet to keep data secret and safe from the attack of unauthorized personnel. As we know World Wide Web in 21st century makes revolution in technology field or to be precise in the Information Technology, and the more revolution are waiting to come. To keep this revolution moving on the Data Mining approach becomes a buzzword in these decades. In a short term data mining refers to approach the practice of examining large pre-existing databases in order to generate new information [26]. A small field of data mining is text mining. Text mining, also referred to as text data mining, roughly equivalent to text analytics, is the process of deriving high-quality information from text [27]. Text mining redirects to one of the important application which is text summarization. Text summarization is one of the challenging and important problems of text mining. It provides several benefits to users. A vast number of real life problems can also come to a fruitful conclusion with text summarization.

In this chapter we will give some idea about text summarizing and its automotive mode. We will also cover the classification of text summarization. Then we will discuss about the objectives of our project. After that, we will cover steps of summarizing process along with brief discussion about extraction based summarizing technique. At the end of this chapter we will try to illustrate why we choose this approach for Bengali Language.

1.1 Text Summarization

Text summarization is way to summarize a single document or multi-document texts and return a concise amount of text a paragraph most likely from where we can have minimum idea about the whole content of the document. There is an enormous amount of textual data and it is rapidly increasing with a blink of eye lead. Those data are unstructured and unorganized. To have a better result and better knowledge and a fair overview about those data is to figure out some way to organize in a specific way. There is a great need to reduce much of this text data to shorter, focused summaries that capture the salient details, both so we can navigate it more effectively as well as check whether the larger documents contain the information that we are looking for

> "*Textual information in the form of digital documents quickly accumulates to huge amounts of data. Most of this large volume of documents is unstructured: it is unrestricted and has not been organized into traditional databases. Processing documents is therefore a perfunctory task, mostly due to the lack of standards* [28]*."*

So the problem appears is we cannot summarize all the text data manually. That's why we need an automated procedure to summarize our document. And that illustrates the idea of an automated text summarizer. The author of a book named "Automatic Text Summarization" published at 2014 provides 6 possible reasons to illustrate the importance of text summarization.

1. Summaries reduce reading time [28].
2. When researching documents, summaries make the selection process easier [28].
3. Automatic summarization improves the effectiveness of indexing [28].
4. Automatic summarization algorithms are less biased than human summarizers [28].
5. Personalized summaries are useful in question-answering systems as they provide personalized information [28].
6. Using automatic or semi-automatic summarization systems enables commercial abstract services to increase the number of texts they are able to process [28].

1.2 Automatic Text Summarization

Ideas about text summarization put a great impact on data scientists. Looking for a decent text summarization approach the idea about an automation process starts expanding its root. An automatic text summarization is a process of a coherent and concise version of a long text document.

> *"Text summarization is the process of distilling the most important information from a source (or sources) to produce an abridged version for a particular user (or users) and task (or tasks) [29]."*

We human being are generally good at this types works that is first reading the whole content, understand the meaning of it and make a concise summary with most possible overview of the content. The goal of automatic text summarizer is to create such a summary that is as good as human. The automatic process is not just generating words or phrases that capture the gist of the source document. It is more like generating a meaningful new document which should be easily readable. Automatic text summarization is the task of producing a concise and fluent summary while preserving key information content and overall meaning [30].

1.3 Classification of Text Summarization

Text summarization is process of transforming a large collection of text documents into reduced and compact text document that represents the digest of the original text data. A summarized document helps in understanding the gist idea of the document instead of reading the whole content. Summarizing certainly is a great concept to reduce the hassles of reading a long textual content and it saves a lot of time.

Automatic text summarization is such a process of generating summaries with the help of a gently written computer program. Texts summarizing has three main steps. These steps are identifying the topic, interpretation, and last step is summary generation [32]. Automatic text summarization can be categorized based on several criteria. The different dimension of text summarization can be generally categorized based on its input type, purpose and output types [31]. There is also another type of classification which is based on uses of external resources.

According to input type there are two types of summarization process. Those are –

i. Single Document: Single document summarization produces summary of single input document. The processing relatively easier. Many of the early summarization system dealt with single document summarization [31].
ii. Multi Document: Multi document summarization produces summary of multiple documents. These multiple inputs are often documents discussing the same topic [31].

According to the purpose there are three types of summarization process. Those are –

i. Generic: Generic summarization purpose is to summarize all texts regardless of its topic or domain; i.e., generic summaries make no assumptions about the domain of its source information and view all documents as homogenous

texts. The majority of the work that has been done revolves around generic summarization [32].

ii. Domain Specific: There have also been developments of summarization systems which are centered upon various domain of interest. For example, summarizing finance articles, biomedical documents, weather news, terrorist events and many more (Radev and McKeown, 1998; Verma et al., 2007; Wu and Liu, 2003). Often, this type of summarization requires domain specific knowledge bases to assist its sentence selection process [31].

iii. Query based: Query-based summary contains only information which is queried by the user. The queries are typically natural language questions or keywords that are related to a particular subject. For instance, snippets produced by search engines are an example of query-based application [31].

According to output types the summarization process is two types. Those are –

i. Abstraction based: Abstraction based summarization produces an abstract of the original text by using interpretation procedure and generate summary that express the same in more concise way.

ii. Extraction based: Extractive summaries or extracts are produced by identifying important sentences which are directly selected from the document. Most of the summarization systems that have been developed are for extractive type summaries (Aliguliyev, 2009; Ko and Seo, 2008). In abstractive summarization, the selected document sentences are combined coherently and compressed to exclude unimportant sections of the sentences (Ganesan et al.,2010; Khan et al., 2015) [31] .In this approach sentences are given some score bases on different criteria and then the sentences with relatively maximum rating are chosen to pick for summarization. It uses several types of Natural Language Process (NLP) to retrieve information.

In this report we will discuss about extractive summarization using two famous approaches namely – Word Counting based preferences and Sentence similarity approach.

According to uses of external resources there are two different types of summarization procedure –

i. Knowledge poor: Knowledge poor summarizer do not use external corpus like Wikipedia, WordNet.

ii. Knowledge rich: Knowledge rich summarizer uses external corpus like Wikipedia, WordNet.

The complete classifications can be shown with a diagram given below –

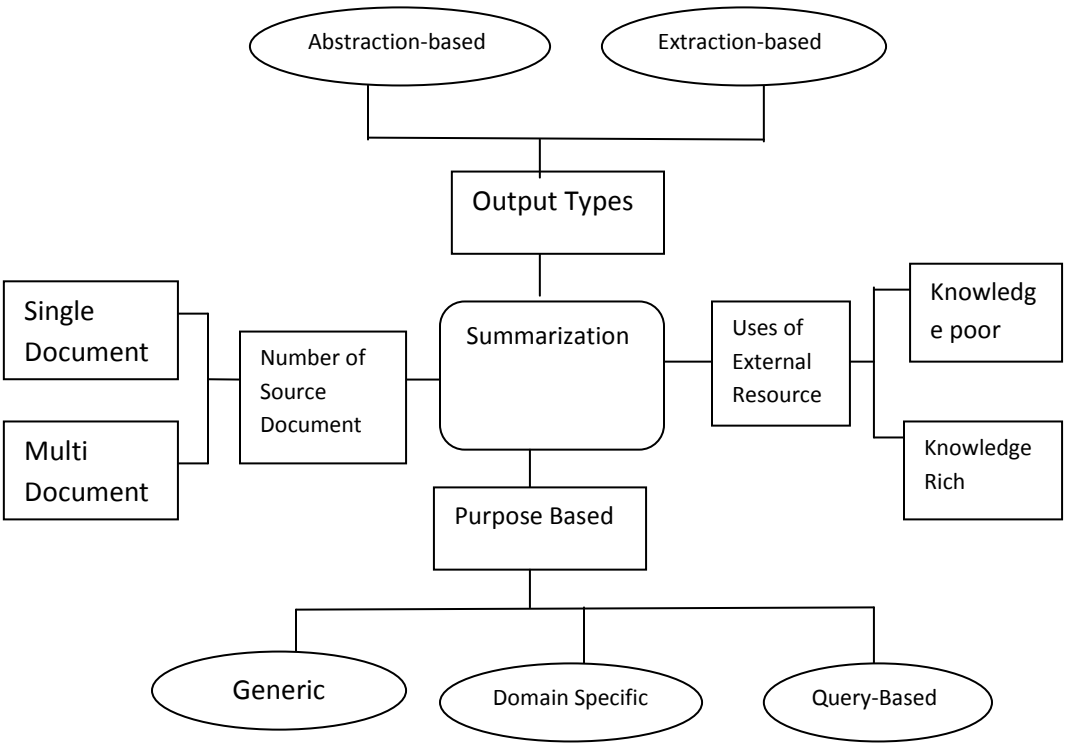

Figure 1: Classification of summarization tasks.

This diagram shown above is able to give a complete overview of text summarization procedure that the researchers tried to figure out and became succeeded with several procedures in the last decades. The classification overview clear illustrates that the through research is still going on this finest part of text mining to make summarization more and more accurate.

1.4 Objectives

There are basically two objectives why I have chosen this project. The first one is to provide summaries of a single document based on extraction summarization technique for several input texts and analyze the results between to come up into a fair and gentle conclusion. The other one is to make an easy to use python text summarizing module for Bengali language to extent a better way for doing more text mining research in our mother tongue.

1.5 Fundamental Steps for Text Summarization

The fundamental steps for summarizing a document is given below –

i. **Topic identification**: The very first step is topic identification. The algorithm that designed based several aspects point out some ways to at least to have an idea about which topic we are working on. The techniques of topic identification include position, Cue phrases, Headline, Tagline, Word frequency. Methods that are centered on the position of phrases are the most useful method in topic identification.

ii. **Interpretation**: In this step the interpretation of the whole document is processed to produce a new content that is summarized with almost maximum amount of information from the main content. The most relevant lines or sentences are figured out to form a summarized document.

iii. **Summary generation**: The automated system then generate summary in this step. The final summary stored as a new raw document.

1.6 Extractive Text Summarization

Extractive text summarization involves the selection of phrases and sentences from the source document to make up the new summary. Techniques involve ranking the relevance of phrases in order to choose only those most relevant to the meaning of the source. Extractive summarization methods work by identifying important sections of the text and generating them verbatim; thus, they depend only on extraction of sentences from the original text [5]. Most of the current automated text summarization systems use extraction method to produce a summary. Sentence extraction techniques are commonly used to produce extraction summaries. One of the methods to obtain suitable sentences is to assign some numerical measure of a sentence for the summary called sentence scoring and then select the best sentences to form document summary based on the compression rate. In the extraction method, compression rate is an important factor used to define the ratio between the length of the summary and the source text. As the compression rate increases, the summary will be larger, and more insignificant content is contained. While the compression rate decreases the summary to be short, more information is lost. In fact, when the compression rate is 5-30%, the quality of summary is acceptable. The most common methods of extractive text summarizing are as follows-

i. Statistical approach
ii. Graph based sentence similarity approach
iii. Machine learning approach
iv. Clustering methods
v. Meta heuristic search approaches.
vi. Fuzzy logic based approach.

In our project, we will illustrate the statistical approach and graph based sentence similarity approach to summarize a given text and to compare their complexity and accuracy. In statistical approach the importance of each sentence is evaluated by the appearances of words i.e. the word frequency along with number of sentences hold those words. In graph based sentence similarity approach every sentence in the content considered as a single node removing duplicates to eliminate redundancy. The connection between each sentence has been done with a certain weight value calculated based on several criteria. This weight actually denotes the similarity values between any two sentences. Then important sentences are sieved out based higher similarity values compared to other sentences.

1.7 Why Bengali?

Before illustrating this issue I would like to remember the national heroes whose sacrifice made this possible for us to speak in Bengali. Think of a situation, a man is sitting outside with coffee cup and a newspaper in his hand. Well, this situation is very much common in our sub-continent. But in outer countries like US, with the vast increasing of technological revolution people become more technical. They are thinking of some other revolution in the budgeted time of reading newspaper. To keep the idea about what is happening they just have a look over the news and given instruction to machine to figure out its gist. So to make these things possible in our mother tongue can also be a great adventure in my point of view. People will do their other works after giving instructions to machine about what's going on the current world. And also as we believe text summarization has many more technical effects on modern day textual mining strategy. So we think why not we just extent a way of these strategies in Bengali.

Chapter 2

Chapter 2 : Background and Literature Review

The **Natural Language Toolkit**, or more commonly **NLTK**, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. It was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania NLTK includes graphical demonstrations and sample data. It is accompanied by a book that explains the underlying concepts behind the language processing tasks supported by the toolkit, plus a cookbook. NLTK is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning. NLTK supports classification tokenization, stemming, tagging , parsing and semantic reasoning functionalities.

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

NLTK has been called “a wonderful tool for teaching, and working in, computational linguistics using Python,” and “an amazing library to play with natural language.”

Natural Language Processing with Python provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analyzing linguistic structure, and more.

2.1 Stop Words

Stop Words are words which do not contain important significance to be used in Search Queries. Usually these words are filtered out from search queries because they return vast amount of unnecessary information. Most Search Engines do not consider extremely common words in order to save disk space or to speed up search results. These filtered words are known as 'Stop Words'. We can access stop word using python NLTK corpus. Before accessing the stop word library we need to get access into the module and to that we have to import the *stopword* module from the corpus of NLTK.

```
from nltk.corpus import stopwords
```

Here is the list of stop words is stored in python's stop word module for English language:

```
>>> set(stopwords.words('english'))
```

```
{'ourselves', 'hers', 'between', 'yourself', 'but', 'again', 'there', 'about', 'once', 'during', 'out',
'very', 'having', 'with', 'they', 'own', 'an', 'be', 'some', 'for', 'do', 'its', 'yours', 'such', 'into',
'of', 'most', 'itself', 'other', 'off', 'is', 's', 'am', 'or', 'who', 'as', 'from', 'him', 'each', 'the',
'themselves', 'until', 'below', 'are', 'we', 'these', 'your', 'his', 'through', 'don', 'nor', 'me',
'were', 'her', 'more', 'himself', 'this', 'down', 'should', 'our', 'their', 'while', 'above', 'both',
'up', 'to', 'ours', 'had', 'she', 'all', 'no', 'when', 'at', 'any', 'before', 'them', 'same', 'and', 'been',
'have', 'in', 'will', 'on', 'does', 'yourselves', 'then', 'that', 'because', 'what', 'over', 'why', 'so',
'can', 'did', 'not', 'now', 'under', 'he', 'you', 'herself', 'has', 'just', 'where', 'too', 'only',
'myself', 'which', 'those', 'i', 'after', 'few', 'whom', 't', 'being', 'if', 'theirs', 'my', 'against', 'a',
'by', 'doing', 'it', 'how', 'further', 'was', 'here', 'than'}
```

To eliminate stop words from certain content in any other language we have to make a similar stop words library in the language as we want. We used utf-8 encoding technique and that's why all the words are will be encoded according this utf-8 encoding procedure. Later while processing the data in our opted language we worked with the encoded content. For our project purpose we made stop word library in Bengali that contains stop words that are commonly used in Bangla language.

2.2 Tokenization

Tokenization is the process of demarcating and possibly classifying sections of a string of input characters. The resulting tokens are then passed on to some other form of processing. The process can be considered a sub-task of parsing input. In this project we have used two types of tokenization:-

(i) Sentence tokenization
(ii) Word tokenization

2.2.1 Sentence Tokenization

String tokenization is the process of extracts all the sentences from a text. Here is the procedure of sentence tokenization:

```
import nltk

sent_text = nltk.sent_tokenize(text) # this gives us a list of sentences
```

The usual sentence tokenization procedure tokenize sentences based on periods (.), semi-colon (;), question mark (?) and exclamation. But in Bengali language the period notation (.) is called Dari and denoted as (।). So to make sure that period in Bengali working properly we made our own sentence tokenize method and manage way to read the end characters as Dari.

2.2.2 Word Tokenization

After sentence tokenization every word is extracted from every sentence. Here is the procedure of word tokenization:

```
import nltk

word_lst = nltk.sent_tokenize(sentence) # this gives us a list of words
```

For all the language over the world the word tokenize method works as same as base method. That's why we decided not to make any changes.

2.3 Similarity between Sentences

Recent applications of natural language processing present a need for an effective method to compute the similarity between very short texts or sentences [25]. Some applications of sentence similarity are text summarization, text categorization etc.

Traditionally, techniques for detecting similarity between long texts (documents) have centered on analyzing shared words. Such methods are usually effective when dealing with long texts because similar long texts will usually contain a degree of co-occurring words. However, in short texts, word co-occurrence may be rare or even null. This is mainly due to the inherent flexibility of natural language enabling people to express similar meanings using quite different sentences in terms of structure and word content. Since such surface information in short texts is very limited, since such surface information in short texts is very limited, this problem poses a difficult computational challenge. The focus of this paper is on computing the similarity between shortest sentences.

Sentence similarity methods can be categorized into 3 categories: *Word co-occurrence methods, Corpus based methods, descriptive based methods*.

The **word co-occurrence** methods are often known as the "bag of words" method. They are commonly used in Information Retrieval (IR) systems [24]. The systems have a precompiled word list with n words. The value of n is generally in the thousands or hundreds of thousands in order to include all meaningful words in a natural language. Each document is represented using these words as a vector in n-dimensional space. A query is also considered as a document. The relevant documents are then retrieved based on the similarity between the query vector and the document vector. This technique relies on the assumption that more similar documents share more of the same words. If this technique were applied to sentence similarity, it would have three obvious drawbacks:

i. The sentence representation is not very efficient. The vector dimension n is very large compared to the number of words in a sentence, thus the resulting number of words in a sentence, thus the resulting vectors would have many null components.
ii. The word set in IR systems usually excludes function words such as the, of, an, etc. Function words are not very helpful for computing document similarity, but cannot be ignored for sentence similarity because they carry structural information, which is useful in interpreting sentence meaning. If function words were included, the value for n would be greater still.

One extension of word co-occurrence methods is the use of a lexical dictionary to compute the similarity of a pair of words taken from the two sentences that are being compared (where one word is taken from each sentence to form a pair). Sentence similarity is simply obtained by aggregating similarity values of all word pairs [28]. Another extension of word co-occurrence techniques leads to the pattern matching methods which are commonly used in conversational agents and text mining [7]. Pattern matching differs from pure word co-occurrence methods by incorporating local structural information about words in the predicated sentences. A meaning is conveyed in a limited set of patterns, where each is represented using a regular expression [14] (generally consisting of parts of words and various wildcards) to provide generalization. Similarity is calculated using a simple pattern matching algorithm. This technique requires a complete pattern set for each meaning in order to avoid ambiguity and mismatches. Manual compilation is an immensely arduous and tedious task. At present, it is not possible to prove that a pattern set is complete and, thus, there is no automatic method for compiling such a pattern set. Finally, once the pattern sets are defined, the

algorithm is unable to cope with unplanned novel utterances from human users. One recent active field of research that contributes to sentence similarity computation is the methods based on statistical information of words in a huge corpus. Well known methods in corpus-based similarity are the latent semantic analysis (LSA) [10], [17], [18] and the Hyperspace Analogues to Language (HAL) model [5]. Some leading researchers in LSA boldly claim that LSA is a complete model of language understanding [17]. In LSA, a set of representative words needs to be identified from a large number of contexts (each described by a corpus). A word by context matrix is formed based on the presence of words in contexts. The matrix is decomposed by singular value decomposition (SVD) into the product of three other matrices, including the diagonal matrix of singular values [19]. The diagonal singular matrix is truncated by deleting small singular values. In this way, the dimensionality is reduced. The original word by context matrix is then reconstructed from the reduced dimensional space. Through the process of decomposition and reconstruction, LSA acquires word knowledge that spreads in contexts. When LSA is used to compute sentence similarity, a vector for each sentence is formed in the reduced dimension space; similarity is then measured by computing the similarity of these two vectors [10]. Because of the computational limit of SVD, the dimension size of the word by context matrix is limited to several hundred. As the input sentences may be from an unconstrained domain (and thus not represented in the contexts), some important words from the input sentences may not be included in the LSA dimension space. Second, the dimension is fixed and, so, the vector is fixed and is thus likely to be a very sparse representation of a short text such as a sentence. Like other methods, LSA ignores any syntactic information from the two sentences being compared and is understood to be more appropriate for larger texts than the sentences dealt with in this work [18]. Another important work in corpus-based methods is Hyperspace Analogues to Language (HAL) [5]. Indeed, HAL is closely related to LSA and they both capture the meaning of a word or text using lexical co-occurrence information. Unlike LSA, which builds an information matrix of words by text units of paragraphs or documents, HAL builds a word-by-word matrix based on word co-occurrences within a moving window of a predefined width. The window (typically with a width of 10 words) moves over the entire text of the corpus. An NXN matrix is formed for a given vocabulary of N words. Each entry of the matrix records the (weighted) word co-occurrences within the window moving through the entire corpus. The meaning of a word is then represented as a 2N-dimensional vector by combining the corresponding row and column in the matrix. Subsequently, a sentence vector is formed by adding together the word vectors for all words in the sentence. Similarity between two sentences is calculated using a metric such as Euclidean distance. However, the authors' experimental results showed that HAL was not as promising as LSA in the

computation of similarity for short texts [5]. HAL's drawback may be due to the building of the memory matrix and its approach to forming sentence vectors: The word-by-word matrix does not capture sentence meaning well and the sentence vector becomes diluted as a large number of words are added to it. The third category of related work is the descriptive features-based methods. The feature vector method tries to represent a sentence using a set of predefined features [22]. Basically, a word in a sentence is represented using semantic features, for example, nouns may have features such as HUMAN (with value of human or nonhuman), SOFTNESS (soft or hard), and POINTNESS (pointed or rounded). A variation of feature vector methods is the introduction of primary features and composite features [12], [13]. Primary features are those primitive features that compare single items from each text unit. Composite features are the combination of pairs of primitive features. A text is then represented in a vector consisting of values of primary features and composite features. Similarity between two texts is obtained through a trained classifier. The difficulties for this method lie in the definition of effective features and in automatically obtaining values for features from a sentence. The preparation of a training vector set could be an impractical, tedious, and time-consuming task. Moreover, features can be well-defined for concrete concepts; however, it still is problematic to define features for abstract concepts. Overall, the aforementioned methods compute similarity according to the co-occurring words in the texts and ignore syntactic information. They work well for long texts because long texts have adequate information (i.e., they have a sufficient number of co-occurring words) for manipulation by a computational method. The proposed algorithm addresses the limitations of these existing methods by forming the word vector dynamically based entirely on the words in the compared sentences. The dimension of our vector is not fixed but varies with the sentence pair and, so, it is far more computationally efficient than existing methods. Our algorithm also considers word order, which is a further aspect of primary syntactic information [1].

2.4 Text Similarity Method

The proposed method derives text similarity from semantic and syntactic information contained in the compared texts. A text is considered to be a sequence of words each of which carries useful information. The words, along with their combination structure, make a text convey a specific meaning. Texts considered in this paper are assumed to be of sentence length. Fig. 2 shows the procedure for computing the sentence similarity between two candidate sentences. Unlike existing methods that use a fixed set of vocabulary, the proposed method dynamically forms a joint word set only using all the

distinct words in the pair of sentences. For each sentence, a raw semantic vector is derived with the assistance of a lexical database. A word order vector is formed for each sentence, again using information from the lexical database. Since each word in a sentence contributes differently to the meaning of the whole sentence, the significance of a word is weighted by using information content derived from a corpus. By combining the raw semantic vector with information content from the corpus, a semantic vector is obtained for each of the two sentences. Semantic similarity is computed based on the two semantic vectors. An order similarity is calculated using the two order vectors. Finally, the sentence similarity is derived by combining semantic similarity and order similarity. The following sections present a detailed description of each of the above steps. Since semantic similarity between words is used both in deriving sentence semantic similarity and word order similarity, we will first describe our method for measuring word semantic similarity.

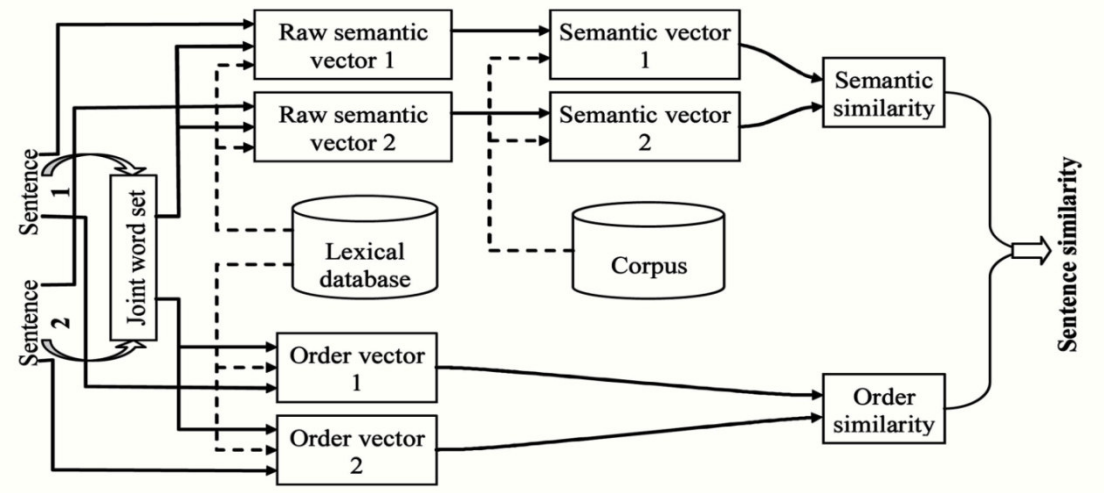

Figure 2: Sentence similarity computation diagram

Chapter 3

Chapter 3 : Implementation of Summarization Techniques

The extractive summarization process evolves with several fascinating algorithms. From those algorithms we worked with the statistical approach and graph based sentence similarity approach. At the very beginning we will show the implementation idea of statistical approach.

3.1 Statistical Approach

As named the statistical approach involved with computing the selection of summarized content using probabilistic method. Here the selected lines for summarization will be chosen from frequency of words and then using simple probability method. In statistical approach we first initialize the min_cut and max_cut values. Words that have probabilistic appearance less than min_cut and more than max_cut will be ignored. The list of stop words then will also be considered as non-countable. After tokenizing every word except the stop words we calculated frequency of each word. In this situation all the words are mapped using python's hashmap structure that stores the words as an object. Then we did calculate the maximum among frequency of words. Then to figure out the statistical score of words we will use the formula given below –

$$W_i = \frac{A_i}{M} \dots\dots\dots\dots (1)$$

Here, A_i is the number of appearances of a certain word obviously without stop words and M is the number of maximum among frequency of words. Doing this calculation for each of the sentences we got the statistical score of each word. After this calculation we have calculated the total score of a sentence by placing the score of all the words and for the stop words it the score is null. The equation of the score of a sentence can be shown as –

$$S_i = \sum_{i=1}^{k} W_k \dots\dots\dots\dots (2)$$

By implementation of this equation once we have the score of all the sentences we can give output the sentences with most scored K sentences at our ease. As we are implanting this algorithm in Bengali we need to make a different library of Bengali stop

words. We also need to make a way to tokenize the sentences as Bengali language does not contains the period used to separate a line. Given below the a snippet that illustrates the way to tokenize the sentences –

```
# Sentences Tokenizer Custom

stcs = [];
tmpstr = "";
words = word_tokenize(text)
for w in words:
  if (w == "।" or w == "?" or w == ";" or w=='!'):
    tmpstr = tmpstr + w;
    stcs.append(tmpstr);
    tmpstr = "";
  else:
    tmpstr = tmpstr + w + " ";
sents = stcs

# Sentences Tokenizer Custom
```

The next snippet will return a list of Bengali stop words from the Bengali stop words library.

```
print(set(get_stop_words('bn')))
```

The Bengali stop words are –

{'প্রাথমিক', 'যাঁরা', 'এস', 'বলল', 'যদিও', 'হোক', 'তার', 'নিয়ে', 'হওয়ার', 'এমনি', 'নয়', 'সঙ্গে', 'ইহা', 'ওঁর', 'আপনার', 'মধ্যভাগে', 'অবধি', 'একে', 'ছাড়া', 'প্রায়', 'সেটা', 'এরা', 'পরেও', 'পরে', 'করছেন', 'ও', 'তারে', 'ই', 'এর', 'এত', 'তবু', 'পর', 'এসে', 'এঁদের', 'চলে', 'ইত্যাদি', 'হলেই', 'অনেক', 'তা', 'ওঁরা', 'মাত্র', 'বা', 'অনেকেই', 'অন্য', 'এ', 'বরং', 'ওরা', 'উপরে', 'করাই', 'হবেন', 'অতএব', 'তারপর', 'স্পষ্ট', 'সেটি', 'ধামার', 'পারে', 'ব্যাপারে', 'ওকে', 'মধ্যেই', 'আগেই', 'নানা', 'ছাড়াও', 'হলেও', 'কে', 'যেতে', 'যাকে', 'জনকে', 'থেকেও', 'বসে', 'কি', 'মোটেই', 'দেন', 'গিয়েছে', 'নিজেদের', 'ফিরে', 'তারা', 'ভাবে', 'যেন', 'বিষয়টি', 'কেথা', 'আমরা', 'যখন', 'বার', 'গেলে', 'গেছে', 'কোটি', 'ফলে', 'শুরু', 'পারেন', 'তোমার', 'হয়েই', 'পক্ষে', 'পর্যন্ত', 'প্রযন্ত', 'নাকি', 'জে', 'হিসাবে', 'তাহার', 'উত্তর', 'করি', 'তিনি', 'করেন', 'এমন', 'তিনঐ', 'আজ', 'নিতে', 'মতোই', 'পারি', 'তাহাতে', 'গুলি', 'নাই', 'একটি', 'হয়', 'তাহলে', 'ঠিক', 'এল', 'এতে', 'করেছেন', 'হইবে', 'দিকে', 'অন্তত', 'কাউকে', 'পি', 'অনুযায়ী', 'জানায়', 'যেমন', 'হওয়ায়', 'পেয়', 'বিনা', 'নিজের', 'হইতে', 'থাকে', 'উপর', 'করবে', 'করছে', 'এটা', 'নিজে', 'নেওয়া', 'হয়তো', 'তাঁর', 'না', 'র', 'কমনে', 'যিনি', 'সে', 'বদলে', 'এটি', 'তাদের', 'সব', 'এদের', 'গিয়ে', 'ফের', 'হয়েছেন', 'এটাই', 'থাকায়', 'যত', 'অনেকে', 'ওদের', 'যায়', 'আবার', 'যতটা', 'লক্ষ', 'সেখানে', 'হয়', 'তাঁদের', 'পাচ', 'চেষ্টা', 'করার', 'ক্ষেত্রে', 'তাহা', 'সেটাই', 'জনের', 'করা', 'তাঁাহারা', 'বি', 'কখনও', 'কাজে', 'হলে', 'যাওয়ার', 'সেটাও', 'তিনিও', 'কিংবা', 'রেখে', 'নেই', 'বললেন', 'শুধু', 'পরেই', 'আছে', 'করতে', 'ওই', 'ধরে', 'রাখা', 'দিয়ে', 'হলো', 'ছিলেন', 'যাওয়া', 'মধ্যে', 'একবার', 'তথা', 'অবশ্য', 'বলা', 'প্রথম', 'এখানে', 'কেউ', 'যথেষ্ট', 'প্রায়', 'সম্প্রতি', 'হয়ে', 'দুটি', 'নতুন', 'সুতরাং', 'বেশ', 'দেওয়া', 'রয়েছে', 'জানা', 'যাবে', 'বিভিন্ন', 'হতেই', 'করলেন', 'বলে', 'তেমন', 'এই', 'ঐ', 'চার', 'যাচ্ছে', 'হবে', 'থাকেন', 'হয়নি', 'করিয়ে', 'নাগাদ', 'বলেছেন', 'কোনও', 'হতে', 'চালু', 'করে', 'কোনো', 'দুই', 'অথবা', 'অর্থাত', 'যে', 'বেশি', 'কিছুই', 'গিয়ে', 'তাকে', 'বন', 'বহু', 'সমস্ত', 'সবার', 'আরও', 'কত', 'হইয়া', 'মতো', 'যাওয়া', 'যাতে', 'হাজার', 'জানিয়ে', 'পেয়ে', 'পাওয়া', 'প্রভৃতি', 'উনি', 'অথচ', 'কাজ', 'তুমি', 'কয়েক', 'ছিল', 'কেন', 'আদ্যভাগে', 'তখন', 'আমার', 'এবং', 'তাঁকে', 'বাদে', 'খুব', 'প্রতি', 'তত', 'হওয়া', 'ব্যবহার', 'করলে', 'দ্বারা', 'জানানো', 'বিশেষ', 'ওখানে', 'থাকবেন', 'আপনি', 'তো', 'আগামী', 'তাঁরা', 'যা', 'আই', 'এঁরা', 'ধরা', 'মধ্যেও',

'কাছে', 'তবে', 'করেই', 'মোট', 'জন', 'জানিয়েছে', 'কারণ', 'তাই', 'যেখানে', 'যাঁর', 'আমাকে', 'সি', 'থাকা', 'জনজন', 'নেওয়া', 'কয়েক', 'ভাবেই', 'এখনও', 'করেছে', 'এক', 'যার', 'বলতে', 'ওর', 'আমাদের', 'কবে', 'জানতে', 'দিতে', 'দেখতে', 'বক্তব্য', 'দেয়', 'হন', 'এবার', 'নিজেই', 'থেকে', 'যাদের', 'যদি', 'দেওয়া', 'কিন্তু', 'আমি', 'উচিত', 'থাকবে', 'তাতে', 'দিলেন', 'দেওয়ার', 'দুটো', 'যারা', 'সামনে', 'থেকেই', 'জন্যওজে', 'হত', 'দিয়েছেন', 'সহিত', 'রকম', 'করিয়া', 'করেছিলেন', 'দিয়েছে', 'কারও', 'হচ্ছে', 'মনে', 'গোটা', 'সহ', 'সেই', 'কাছ', 'এখানেই', 'এখন', 'তাও', 'তুলে', 'দু', 'হয়েছে', 'দেখা', 'কিছু', 'হয়েছিল', 'নিয়ে', 'দিন', 'টি', 'আর', 'দেখে', 'গেল', 'আগে', 'এমনকী', 'করিতে', 'করায়', 'মাধ্যমে', 'একই', 'এব', 'চায়', 'জন্য', 'এতটাই', 'কয়েকটি', 'কোন', 'সেখান', 'বলেন', 'হল', 'চান', 'সাধারণ', 'হেলে', 'নয়', 'ওঁদের', 'নেওয়ার', 'স্বয়ং', 'কী', 'করবেন', 'চেয়ে', 'কেউই', 'যান', 'সঙ্গেও'}

It is certain that there are more stop words than the list made above. But for analytical purpose we just store a decent collection.

Now we are moving to the implementation of the statistical approach.

```
freq = defaultdict(int)
    for s in word_sent:
      for word in s:
        if word not in self._stopwords:
          freq[word] += 1
    # frequencies normalization and filtering
    m = float(max(freq.values()))

    for w in list(freq.keys()):
      freq[w] = freq[w]/m
      if freq[w] >= self._max_cut or freq[w] <= self._min_cut:
        del freq[w]
    return freq
```

The snippet given above computes the frequency of each of word. As input we pass a list of tokenized words through the method parameter and the as an out the method will return a dictionary that contains the score of individual words. It is an implementation of equation 1 written above.

The second snippet given below will calculate the score of a complete sentence using the scores of words within a single sentence. This is the implementation of the equation 2 written above.

```
word_sent = [word_tokenize(s.lower()) for s in sents]
    self._freq = self._compute_frequencies(word_sent)
    ranking = defaultdict(int)
    for i,sent in enumerate(word_sent):

      for w in sent:
        if w in self._freq:
          ranking[i] += self._freq[w]
    sents_idx = self._rank(ranking, n)

    return [sents[j] for j in sents_idx]
```

As an input it will take a list of sentences and then return the list of *K* sentences based on the score.

And using this way we can summarize text content and the result is somewhat looks promising. As the concept is based on a human thought and as we consider if something is repeating then it much more important than others things. Hence the data seems very much promising.

3.2 Graph based Sentence Similarity Approach

Graph based sentence similarity approach is basically is such an approach every sentence is considered as a single node. Each of the nodes will directly connect to each other. So if there are *N* sentences there will be $\frac{N(N-1)}{2}$ edges after completion of connection. Obviously the connection will be bi-directional. The cost between any two nodes will be similarity between two sentences that will be calculated by semantic similarity. Important sentences with higher similarity are evaluated on the base of which sentence max match with other sentences.

Let us consider four players and their preferences for captaincy. Think of a perfect democratic system, where each player can recommend any number of players with varying degrees of recommendation. Thus the captain is to be determined by a preferential list supplied by each member of the team. Each player can cast a vote varying from 0 to 1 based upon preference. They also have the freedom to assign 0 to any player. Let us assume all the players uniformly name themselves for the captaincy (vote: 1) or totally exclude themselves from consideration (vote: 0).Given the above scenario, which player will be selected as captain? Clearly the player who attains most vote. We adopt the very same approach in picking up the most salient sentences.

That is the very simple idea of picking sentences for this approach.

3.3 Semantic Similarity between Words

WordNet is a lexical semantic network to hold semantic relations like synonyms and word-senses as the nodes of the network and relations of the synonyms and word-sense are the edge of the network. In WordNet meaning of each word is represented by a unique word-sense and a set of synonyms called synset [33].

3.3.1 Pre-Processing

Text preprocessing simply illustrates organize texts for further usage. Pre Processing is vital pre-requisite while working with noisy text content like social media, news content crawler and all that. Pre-processing involves splitting content into valid tokens: words and symbols, stemming, moving out stop words and parts of speech tagging. Here we have used the nltk tokenizer with a slight modification although built for English works smoothly for Bengali as well.

3.3.2 Computation of Word Similarity

Working on the word similarity we have to first illustrate the WordNet lexicon analysis which is a Bengali lexical knowledge base that will obviously help for NLP and next we will illustrate the word similarity procedure.

3.3.2.1. WordNet Lexicon Formation

To calculate word similarity between any two Bengali words a lexicon should be developed. In this we have used the lexicon developed by (Manjira, Sinha, Tithankar, Anupam). In that lexicon they constructed a hierarchical conceptual graph. The storage and organization of their database facilitate computational processing of the information and efficient searching to retrieve the details associated with any words[34] . Therefore, it is really a fantastic resource and tool for NLP analyzing in Bengali. It supports query like DETAILS(X) where X can be any type of node in hierarchy and SIMILARITY (W1, W2) that returns the similarity between any two words based on the word's organization i.e. either they are synonym , similar, stop word etc. The formation of the lexicon is pretty simple. The lexicon contains 30 different sections as 30 different roots. Words are also categorized, sub-categorized using parts of speech. Corresponding each of the category there are two clusters. The first one contains the synonym of a word and the other one contains related word [34]. The image below will illustrate the lexicon basic formation.

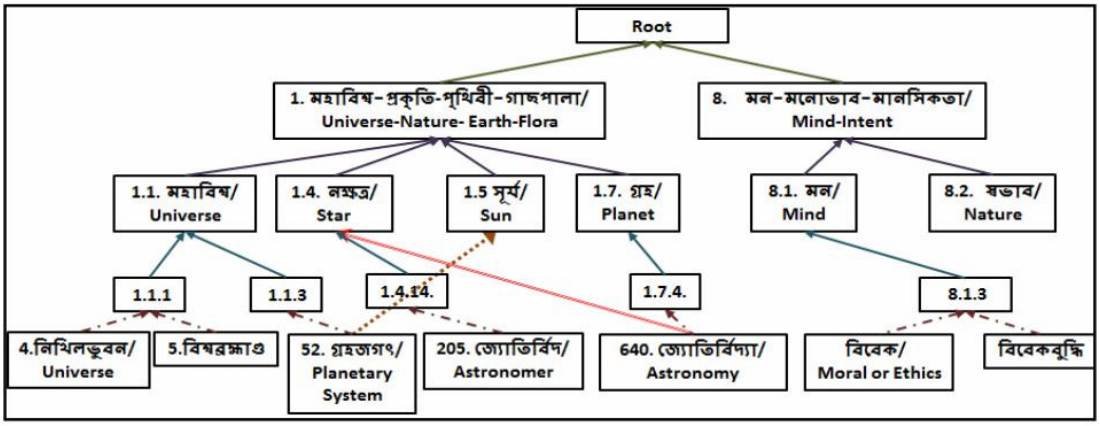

Figure 3: Lexicon formation

This lexicon formation helps to figure out the word similarity. We will illustrate it in the next section.

3.3.2.2 Word Similarity Computation

To calculate word similarity based on the lexicon we will measure the distance between two words. Let us given two words "নিখিলভুবন/Universe" and "বিশ্বব্রহ্মান্ড/Universe". As both of them are in same cluster i.e. within the synonym cluster then the similarity is obviously same. But if we consider another two words "নিখিলভুবন/Universe" and "জ্যোতির্বিদ/Astronomer". Both of them are in different cluster. They are not synonym to each other. Instead they are related word to each other and the distance between those two words are path length which is 4. As we are calculating similarity then we must have to choose the words with more similarity i.e. lesser path distance. However this method may be less accurate if it is applied to larger and for more general semantic nets such as WordNet. For example, let us have the distance between words "মানুষ/Man" and "পশু/Animal" is lesser than the distance between words "মানুষ/Man" and "ডাক্তার/Doctor". Then obviously similarity returns wrong result. To resolve this issue we may create categorized word lexicon. If we create a category called "Human" then put all related words under this category and another category called "Animal" then put all the related words under this category we hope task will be a bit easier and the distance will be a larger such as adding the real distance with a constant value. Therefore, the depth of word in the hierarchy should be taken into account. In summary, similarity between words is determined not only by path lengths but also by depth. We propose that the similarity *S(w1, w2)* between words w1 and w2 is a function of path length and depth as follows:

$$s(w1, w2) = f(l, h) \dots \dots \dots (3)$$

where *l* is the shortest path length between *w1* and *w2*, *h* is the depth of subsume in the hierarchical semantic nets. We assume that (3) can be rewritten using two independent functions as:

$$s(w1, w2) = f1(l)\, f2(h) \ldots\ldots\ldots (4)$$

f1 and *f2* are transfer functions of path length and depth, respectively. We call these information sources, of path length and depth, attributes.

3.3.2.3 Transfer Function and its Characteristics

Transfer functions are those functions that calculate similarities between words or sentences based on several parameters. The values of an attribute in (4) may cover a large range up to infinity, while the interval of similarity should be finite with extremes of exactly the same with a value of 1 and no similarity as 0, then the interval similarity is [0, 1]. Direct use of information sources as a metric of similarity is inappropriate due to it infinite property. Therefore it is intuitive that the transfer function from information sources to semantic similarity is a nonlinear function. Taking path length as an example, when the path length decreases to zero, the similarity would monotonically increase toward the limit 1, while the path length increases infinitely, the similarity should monotonically decreases to 0. So to meet these constraints the transfer function must be a nonlinear function. The nonlinearity of the transfer function is taken into account in the derivation of the formula for semantic similarity between two words, as in the following section.

3.3.2.4 Path Length and its Contribution

Considering Figure 3, the path length between any two words w_1 and w_2 can be determined by one of the following three cases:

1. w_1 and w_2 are in the same synset.
2. w_1 and w_2 are not in the same synset, but the synset for w_1 and w_2 contains one or more common words.
3. w_1 and w_2 are neither in the same synset nor do their synsets contain any common words.

Case 1 simply illustrate that both of the words are of same meaning and hence the path length is 0. For case 2 partially they share the same feature and hence path length is 1.

For case 3, we will compute the actual path length. Based on the previous consideration we set $f1(l)$ in (4) to be a monotonically decreasing function of l :

$$f1(l) = e^{\alpha l} \dots\dots\dots\dots (5)$$

where, α is a constant. The selection of the function in exponential form ensures that $f1$ satisfies the constraint and the value of $f1$ within the range of 0 and 1.

3.2.2.5 Effects of Depth Scaling

Words at upper layers of hierarchical semantic nets have more general concepts and less semantic similarity between words than words at lower layers. This behavior must be taken into account in calculating S(*w1, w2*). We therefore need to scale down S(*w1, w2*) for subsuming words at lower layers. As a result $f2(h)$ should be monotonically increasing with respect to *h*. So the $f2$ will be:

$$f2 = \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \dots\dots\dots\dots . (6)$$

where, β is a smoothing factor. As $\beta \to \infty$ then the depth of a word in the semantic nets is not considered. In summary, the formula for a word similarity measure as:

$$s(w1, w2) = e^{\alpha l} \ \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \dots\dots\dots\dots\dots (7)$$

Where $\alpha \in [0, 1]$ and $\beta \in (0, 1]$ are parameters scaling the contribution of shortest path length and depth, respectively. The optimal values of α and β are dependent on the knowledge base used and can be determined using a set of word pairs with human similarity rating. With several calculations it has been seen that the proposed measure of α and β is 0:2 and 0:45 [20].

3.3 Semantic Similarity between Sentences

A sentence consists of some words. It would be easier to calculate things if we represent a sentence as a vector of words. Given two sentence T1 and T2, a joint word set is formed:

$$T = T1 \ \cup T2$$
$$= \{ w_1, w_2, w_3, \dots\dots, w_m \}$$

The joint word set *T* contains all the distinct word set. Since inflectional morphology may cause a word to appear in a sentence with a different form that conveys a specific meaning for a specific context, we consider them as different words. For example, বালক (boy) and বালকগুলো (boys) considered as two different words and all will be included in the word set. Consider two sentences:

T1 = "সাঈদীর আমৃত্যু কারাদন্ড প্রদান করায় হরতাল ডেকেছে জামায়াত"

T2 = "জামায়াতের হরতাল চলছে সাঈদীর আজীবন কারাদন্ড দেয়ার প্রতিবাদে"

T = { সাঈদীর আমৃত্যু কারাদন্ড প্রদান করায় হরতাল ডেকেছে জামায়াত জামায়াতের চলছে দেয়ার প্রতিবাদে }

Here in *T* the algorithm removes duplicate words skipping the stop words. The vector derived from the joint word set is called the lexical semantic vector, denoted by S. Each entry of the semantic vector corresponds to a word in the joint word set, so the dimension equals the number of words in the joint word set. The value of an entry of the lexical semantic vector, $S_i(i=1,2,....m)$ is determined by the semantic similarity of the corresponding word to a word in the sentence. Take T1 as an example:

Case 1: If w_i appears in the sentence, si is set 1

Case 2: If w_i is not contained in *T1*, a semantic similarity score is computed between w_i and each word in the sentence *T1*, using the method presented in Section 3.2. Thus, the most similar word in *T1* to w_i is that with the highest similarity score ς. If ς exceeds a preset threshold, then $si = \varsigma$ otherwise, $si = 0$.

The reason for the introduction of the threshold is twofold. First, since we use the word similarity of distinct words (different words), the maximum similarity scores may be very low, indicating that the words are highly dissimilar. In this case, we would not want to introduce such noise to the semantic vector. Second, classical word matching methods [1] can be unified into the proposed method by simply setting the threshold equal to one. Unlike classical methods, we also keep all function words. This is because function words carry syntactic information that cannot be ignored if a text is very short (e.g., sentence length). Although function words are retained in the joint word set, they contribute less to the meaning of a sentence than other words. Furthermore, different words contribute toward the meaning of a sentence to differing degrees. Thus, a scheme

is needed to weight each word. We weight the significance of a word using its information content. It has been shown that words that occur with a higher frequency (in a corpus) contain less information than those that occur with lower frequencies [24]. The value of an entry of a semantic vector is:

$$S_i = \tilde{s}.I(w_i).I(\widetilde{w_i}) \dots\dots\dots\dots (8)$$

where, w_i is a word of joint set, $\widetilde{w_i}$ is its associated word in sentence. The use of $I(w_i)$ and $I(\widetilde{w_i})$ allows the concerned two words to contribute to the similarity based on their individual information contents. Here w_i and $I(\widetilde{w_i})$ can be calculated as follows:

$$I(w_i) = 1 - \frac{\log(\mathrm{n}+1)}{\log(\mathrm{N}+1)} \dots\dots\dots\dots (9)$$

Where, n = number of times word w in corpus

N = number of words in the corpus

So $I \in [0, 1]$. The semantic similarity between two sentences is defined as the cosine co-efficient between the two vectors:

$$S_s = \frac{s1.s2}{\|s1\|.\|s2\|} \dots\dots\dots\dots\dots (10)$$

It is worth mentioning that the method does not currently conduct word sense disambiguation for poly-semous words. This is based on the following consideration: First we wanted to our model to be as simple as possible and not too demanding in terms of computing resources. The integration of word sense disambiguation would scale up the model complexity. Second existing sentences similarity methods do not have included word sense.

3.4 Word Order Similarity between Sentences

The word order similarity attempts to correct for the fact that sentences with the same words can have radically different meanings. This is done by computing the word order vector for each sentence and computing a normalized similarity measure between them. The word order vector, like the semantic vector is based on the joint word set. If the word occurs in the sentence, its position in the joint word set is recorded. If not, the similarity to the most similar word in the sentence is recorded if it crosses a threshold η else it is 0. In equations,

$$S_r = \frac{\|r1 - r2\|}{\|r1 + r2\|} \ldots\ldots\ldots\ldots (11)$$

where, $r1$ = word position vector for sentence 1

$r2$ = word position vector for sentence 2

That is, word order similarity is determined by the normalized difference of word order. The following analysis will demonstrate that S_r is an efficient metric for indicating word order similarity. To simplify the analysis, we will consider only a single word order difference, as in sentences *T1* and *T2*. Given two sentences, *T1* and *T2*, where both sentences contain exactly the same words and the only difference is that a pair of words in *T1* appears in the reverse order in *T2*. The word order vectors are:

$$r1 = \{a_1 \ldots a_j \ldots a_{j+k} \ldots a_m\} \text{ for } T_1;$$

$$r2 = \{b_1 \ldots b_j \ldots b_{j+k} \ldots b_m\} \text{ for } T2:$$

a_j and a_{j+k} are the entries for the considered word pair in *T1*, b_j and b_{j+k} are the corresponding entries for the word pair in *T2*, and k is the number of words from w_j to w_{j+k}. From the above assumptions, we have:

$$a_i = b_i = i \text{ for } i = 1, 2, \ldots, m \text{ except } i \neq j, j + k;$$

$$a_j = b_j + k = j,$$

$$a_j + k = b_j = j + k,$$

$$\| r1 \| = \| r2 \| = \| r \|;$$

Then,

$$Sr = 1 - \frac{k}{\sqrt{2\left\|r1\right\|^2 - k^2}} \ldots\ldots\ldots\ldots (12)$$

We can also derive the same formula for a sentence pair with only one different word at the k^{th} entry. For the more general case with a more significant difference in word order or a larger number of different words, the analytical form of the proposed metric

becomes more complicated (which we do not intend to present in this paper). The above analysis shows that the S_r is a suitable indication of word order information. S_r equals 1 if there is no word order difference. S_r is greater than or equal to 0 if word order difference is present. Since S_r is a function of *k*, it can reflect the word order difference and the compactness of a word pair. The following features of the proposed word order metric can also be observed:

1. S_r can reflect the words shared by two sentences.
2. S_r can reflect the order of a pair of the same words in two sentences. It only indicates the word order, while it is invariant regardless of the location of the word pair in an individual sentence.
3. S_r is sensitive to the distance between the two words of the word pair. Its value decreases as the distance increase.
4. For the same number of different words or the same number of word pairs in a different order, S_r is proportional to the sentence length (number of words); its value increases as the sentence length increases. This coincides with intuitive knowledge, that is, two sentences would share more of the same words for a certain number of different words or different word order if the sentence length is longer.

Therefore, the proposed metric is a good one for indicating the word order in terms of word sequence and location in a sentence.

3.5 Sentence Similarity in Total

Similarity between sentences based on semantic nets is basically the lexical similarity of sentences. The relationship between words can be illustrated as word order similarity. This word order similarity illustrates which word will be there in the summarized sentence and which words will be there before which words. The information that is collected in semantic and syntactic way provides information about the meaning of the sentences. The similarity between two sentences is modeled as a linear combination of their semantic similarity and word order similarity. In equations –

$$S(T_1, T_2) = \delta S_s + (1 - \delta) S_r$$
$$= \delta \frac{s1.s2}{\|s1\|.\|s2\|} + (1 - \delta) \frac{\|r1 - r2\|}{\|r1 + r2\|} \dots\dots\dots (13)$$

Here the value of $\delta \leq 1$ decides the relative contributions of semantic and word order information to the overall similarity computation. As syntax plays a sub-ordinate role for semantic processing of text [11], should be a value greater than 0.5, i.e. $\delta \in (0.5, 1]$.

As a short note we would like to mention that, implementing this method requires the use of two databases. The first one is WordNet and the second one is Brown Corpus. WordNet is lexical database of words that returns similarity issue between two words. It's a more like tree hierarchical formation. Starting from the several roots it will move towards leaf in order to find out whether words are in same corpus or not. The algorithm then takes decisions based on this return value. The Brown corpus is also a database of lexical words but it computes the appearance of words within the corpus using relative frequency:

$$p(w) = \frac{n + 1}{N + 1} \dots\dots\dots (14)$$

Here, n = number of times word w occurs in corpus

N = number of words in the corpus

Chapter 4

Chapter 4 : Result and Discussion

In this chapter we will show the result and analysis of both text summarization algorithms that we have used to summarize text using several inputs. We also will show the accuracy based on each input and the relative run time that will help us in further analysis.

4.1 Result and Analysis

We worked 5 different inputs of different size.

Sample Input 1: Size 3KB

ভাত বাঙালির বহুকালের প্রিয় খাদ্য । সরু সাদা চালের গরম ভাতের কদর সবচাইতে বেশি ছিল বলে মনে হয় । পুরোনো সাহিত্যে ভালো খাবারের নমুনা হিসেবে যে-তালিকা দেওয়া হয়েছে, তা হলো কলার পাতায় গরম ভাত, গাওয়া ঘি, নালিতা শাক, মৌরালা মাছ আর খানিকটা দুধ । লাউ, বেগুন ইত্যাদি তরকারি প্রচুর খেত সেকালের বাঙালিরা, কিন্তু ডাল তখনো বোধহয় খেতে শুরু করেনি । মাছ তো প্রিয় বস্তুই ছিল । বিশেষ করে ইলিশ মাছ । শুঁটকির চল সেকালেও ছিল বিশেষ করে দক্ষিণাঞ্চলে । ছাগলের মাংস সবাই খেত । হরিণের মাংস বিয়েবাড়িতে বা এরকম উ□সবে দেখা যেত । পাখির মাংসও তা-ই । সমাজের কিছু লোক শামুক খেত । ক্ষীর, দই, পায়েস, ছানা-এসব ছিল বাঙালির নিত্যপ্রিয় । আম-কাঁঠাল, তাল-নারকেল ছিল প্রিয় ফল । খুব চল ছিল নাড়ু, পিঠেপুলি, বাতাসা, কদমা-এসবের। মসলা-দেওয়া পান পান খেতে সকলে ভালবাসত ।

Sample Output 1

Statistical Approach:

খুব চল ছিল নাড়ু , পিঠেপুলি , বাতাসা , কদমা-এসবের। মসলা-দেওয়া পান পান খেতে সকলে ভালবাসত । মাছ তো প্রিয় বস্তুই ছিল । ভাত বাঙালির বহুকালের প্রিয় খাদ্য । পুরোনো সাহিত্যে ভালো খাবারের নমুনা হিসেবে যে-তালিকা দেওয়া হয়েছে , তা হলো কলার পাতায় গরম ভাত , গাওয়া ঘি , নালিতা শাক , মৌরালা মাছ আর খানিকটা দুধ ।

[Time: 0.058 seconds]

Sentence Similarity Approach:

মাছ তো প্রিয় বস্তুই ছিল । আম-কাঁঠাল, তাল-নারকেল ছিল প্রিয় ফল । বিশেষ করে ইলিশ মাছ । শুঁটকির চল সেকালেও ছিল বিশেষ করে দক্ষিণাঞ্চলে ।

[Time: 13 Seconds]

A Random User:

বাঙালি জাতির জীবনযাত্রার পরিচয়ের মধ্যে খদ্যাভ্যাস অন্যতম। প্রাচীনকাল থেকে এদেশের মানুষ বিচিত্র ধরনের সাধারণ খাবার খেত। উ□সব বা বিয়েতে হরিণের মাংস পরিবেশন করা হতো। সমাজের সকল স্তরের ও অঞ্চলের খাদ্যাভ্যাস প্রায় একই ধরনের ছিল।

[Time: 18 Second]

Analysis: Comparing both statistical and sentence similarity approach we can see that one line is completely same. Then for several other lines we can see that some words are also matched. But from the content both of the summarized document indicates the food habit of some place. Comparing the user's output we can have a idea about our approach is that, at least output of the algorithm's indicate the food habit of Bangladesh. So for this input we can assume that, both algorithms have accuracy about near to 50%.

Sample Input 2: 2 KB

মানুষ সৃষ্টির শ্রেষ্ঠ প্রাণী । জগতের অন্যান্য প্রাণির সহিত মানুষের অয়ার্থক্য - মানুষ বিবেক ও বুদ্ধির অধিকারী । এই বিবেক, বুদ্ধি ও জ্ঞান নাই বলিয়া আর সকল প্রানী মানুষ অপেক্ষা নিকৃষ্ট । জ্ঞান ও মনুষ্যত্বের উতকর্ষ সাধর করিয়া মানুষ জগতের বুকে অক্ষয়কীর্তি স্থাপন করিয়াছে, জগতের কল্যাণ সাধন করিতেছে, পশুবল ও অর্থবল মানুষকে বড় বা মহ□ করিতে পারে না । মানুষ বড় হয় জ্ঞান ও মনুষ্যত্বের বিকাশে । জ্ঞান ও মনুষ্যত্বের প্রকৃত বিকাশে জাতির জীবন উন্নত । প্রকৃত মানুষই জাটিয় জীবনের প্রতিষ্ঠা ও উন্নয়ন আনয়নে সক্ষম ।

Sample Output 2

Statistical Approach:

জ্ঞান ও মনুষ্যত্বের উতকর্ষ সাধর করিয়া মানুষ জগতের বুকে অক্ষয়কীর্তি স্থাপন করিয়াছে , জগতের কল্যাণ সাধন করিতেছে , পশুবল ও অর্থবল মানুষকে বড় বা মহ□ করিতে পারে না ।

[Time: 0.058 Seconds]

Sentence Similarity Approach:

মানুষ বড় হয় জ্ঞান ও মনুষ্যত্বের বিকাশে ।

[Time: 10 Seconds]

A Random User:

সৃষ্টির শ্রেষ্ঠ প্রানী হিসেবে জ্ঞান ও মনুষ্যত্বের গুনে মানুষ জগতে যে অমরকীর্তি গড়ে তুলেছে পশুবল ও অর্থবল দিয়ে তা কখনো সম্ভব নয় ।

[Time: 7 Second]

Analysis: Comparing both of the algorithm it is clear that both output illustrates greatness of human being. If we take a look at the random user's output it also points to

the same direction. Analyzing just about the output we can say that both algorithm's accuracy is about 80%.

Sample Input 3: 4 KB

শিক্ষা বা জ্ঞান অর্জন হলো সাধনার ব্যাপার । তবে এই সাধনার সাধক হতে হবে শিষ্যের নিজেকেই । একজনের সাধনা কখনও অন্য কেউ করে দিতে পারে না। যার সাধনা তাকেই সাধন করতে হয় । অন্যথায় সাধনার ফলাফল কখনই আশানুরূপ হয় না। আমাদের অনেকের মধ্যেই একটি বিশেষ প্রবণতা লক্ষ্য করা যায়, তা হলো গুরু বা শিক্ষকের উপর সম্পূর্ণ ভরসা করে বসে থাকা । আমাদের এই প্রবণতার কারণেই আমাদের শিক্ষা শতভাগ পরিপূর্ণ হয় না । গুরু কিংবা শিক্ষক নিঃসন্দেহে একজন ছাত্রের নিকট ভরসার পাত্র হবে এটাই স্বাভাবিক । কিন্তু তার মানে এই নয় যে, গুরুই তার শিক্ষাকে অন্তরে গেঁথে দেবেন। অন্তরে গেঁথে দেওয়ার দায়িত্ব গুরুর নয় । গেঁথে নেওয়ার দায়িত্ব শিষ্যের । গুরু বড়জোর পথ দেখিয়ে দিতে পারেন । গুরু শুধুমাত্র শিষ্যকে বলে দিতে পারেন কোন পথ তার জন্য উত্তম, কোন পথে, কিভাবে হেটে গেলে সে তার কাঙ্ক্ষিত বস্তুর দেখা পেতে পারে । কিন্তু এরপরের সব দায়িত্বই শিষ্যের । গুরুর দেখানো পথে, গুরুর নির্দেশিত পন্থায় হেটে যেতে হবে শিষ্যের নিজেকেই । সঠিকভাবে সে পথ পাড়ি দিয়ে কাঙ্ক্ষিত বস্তুটি অর্জন করে আনা শিষ্যেরই দায়িত্ব । আমরা প্রায়ই ছাত্রের খারাপ ফলাফলের জন্যই শিক্ষককেই দোষারোপ করি । কিন্তু খারাপ ফলাফলের জন্য কখনও শিক্ষক দায়ী নয়, বরং ছাত্ররাই দায়ী । কিন্তু শিক্ষকের দেখানো পথে ছাত্র যদি হাঁটতে না পারে সে অযোগ্যতা শুধুমাত্র ছাত্রের । শিষ্য পথভ্রষ্ট হলে সে ত্রুটির ভার শিষ্যকেই বহন করতে হয় । সে ভার গুরুর উপর চাপিয়ে দিলে তা কখনও সুবিচার হয় না । গুরু শিষ্যের মঙ্গল কামনা করেন এবং কল্যাণের পথই দেখিয়ে থাকেন । কিন্তু সেই কল্যাণ সাধনে যদি শিষ্যের সাধনায় ত্রুটি থাকে, তবে তা একান্তই শিষ্যের অপারগতা ।

Sample Output 3

Statistical Approach:

গুরু শুধুমাত্র শিষ্যকে বলে দিতে পারেন কোন পথ তার জন্য উত্তম , কোন পথে , কিভাবে হেটে গেলে সে তার কাঙ্ক্ষিত বস্তুর দেখা পেতে পারে । গুরুর দেখানো পথে , গুরুর নির্দেশিত পন্থায় হেটে যেতে হবে শিষ্যের নিজেকেই । কিন্তু সেই কল্যাণ সাধনে যদি শিষ্যের সাধনায় ত্রুটি থাকে , তবে তা একান্তই শিষ্যের অপারগতা । কিন্তু তার মানে এই নয় যে , গুরুই তার শিক্ষাকে অন্তরে গেঁথে দেবেন। অন্তরে গেঁথে দেওয়ার দায়িত্ব গুরুর নয় ।

[Time 0.066 Seccond]

Sentence Similarity Approach:

গুরু বড়জোর পথ দেখিয়ে দিতে পারেন । গুরু শুধুমাত্র শিষ্যকে বলে দিতে পারেন কোন পথ তার জন্য উত্তম, কোন পথে, কিভাবে হেটে গেলে সে তার কাঙ্ক্ষিত বস্তুর দেখা পেতে পারে । তবে এই সাধনার সাধক হতে হবে শিষ্যের নিজেকেই । গুরুর দেখানো পথে, গুরুর নির্দেশিত পন্থায় হেটে যেতে হবে শিষ্যের নিজেকেই ।

[Time: 22 Seconds]

A Random User:

বিদ্যার সাধনা শিষ্যকে নিজে অর্জন করতে হয়, গুরু উত্তরসাধক মাত্র । গুরু কেবল উত্তম পথ দর্শ করান । কিন্তু সেপথে সাধনে

করে সিদ্ধি লাভ করতে শিষ্যকেই ।

[Time: 25 Second]

Analysis: Comparing both approaches's output we can clearly understand that both content is talking about the greatness of teacher and duties of student and also how to show respect to teachers. A sentence is completely same in both of the content. And maximum words are also similar. Comparing to the random user's output it is clear that all 3 content are talking about same thing. And the input paragraph is also talking about this issue. We can take accuracy 80%.

Sample Input 4: 11 KB

তাঁর নামটা শুনলেই চোখের সামনে ভেসে ওঠে ব্রাজিল দলের রক্ষণভাগে ডান প্রান্ত দিয়ে এক অক্লান্ত ফুটবলারের সমানতালে রক্ষণ ও আক্রমণ । খেলেছেন ব্রাজিলের ইতিহাসে সবচেয়ে বেশি ম্যাচ । কপালে একমাত্র খেলোয়াড় হিসেবে টানা তিন বিশ্বকাপের ফাইনাল খেলার গৌরবতিলক । তিনি মার্কোস ইভানগেলিস্তা দি মোরাইস । চিনতে পারলেন না বুঝি ? ফুটবলবিশ্ব অবশ্য তাঁকে কাফু নামেই চেনে । রাশিয়া বিশ্বকাপের প্রাক্কালে বিশ্বজয়ী সাবেক এই রাইটব্যাকের কাঁধে ব্রাজিলের সর্বকালের সেরা একাদশ নির্বাচনের গুরুদায়িত্ব অর্পণ করেছিল ইংলিশ সংবাদমাধ্যম 'দ্য গার্ডিয়ান' । ভীষণ কঠিন এই কাজটা করতে গিয়ে কাফু কাকে কাকে রেখেছেন তাঁর ব্রাজিল একাদশে ? কাকেই বা বাদ দিয়েছেন ? আসুন জেনে নেই—

৪-৪-২ ছকে কাফু যে একাদশ গঠন করেছেন, তাঁর ভাষায় সবাই অতি-আক্রমণাত্মক । 'আমরা হয়তো এই স্কোয়াড খেলালে অনেক গোল খাব, কিন্তু এটাও নিশ্চিত, গোল দিতে পারব তার থেকেও বেশি!'—নিজের একাদশ নিয়ে কাফুর ব্যাখ্যা! গোলরক্ষক হিসেবে কাফুর একাদশে স্থান করে নিয়েছেন ক্লদিও তাফারেল । 'সেলেসাও' গোলপোস্টের নিচে সবচেয়ে বেশি (১০১) ম্যাচ খেলা সাবেক গোলরক্ষক । ১৯৯০, ১৯৯৪ ও ১৯৯৮—তিনটি বিশ্বকাপ খেলা তাফারেলকে দলে রাখতে গিয়ে কাফু বাদ দিয়েছেন একসময়ের সতীর্থ দিদা, ১৯৫৮ ও ১৯৬২ সালের বিশ্বকাপজয়ী গোলরক্ষক জিলমার এবং ১৯৭০ বিশ্বকাপের এমারসন লিয়াওকে । তাফারেলের অন্তর্ভুক্তি প্রসঙ্গে কাফুর ব্যাখ্যা, মাঠে সে আহামরি কিছু করতে চাইত না, এতে বলটা তাঁর নিয়ন্ত্রনেই থাকত। পেছন থেকে ঠান্ডা মাথায় নির্দেশনা দিয়ে যেত সতীর্থদের । চারজনের রক্ষণভাগে রাইটব্যাক হিসেবে কাফু নিজেকে বিবেচনা করেননি । বেছে নিয়েছেন, বিশ্বকাপের ইতিহাসে সর্বকালের অন্যতম সেরা দল হিসেবে খ্যাত '৭০-এর ব্রাজিল দলের অধিনায়ক—কার্লোস আলবার্তো তোরেসকে । সেই বিশ্বকাপের ফাইনালে পেলের পাস থেকেবুলেট গতির শটে অনিন্দ্যসুন্দর একটি গোল করেছিলেন আলবার্তো । বল পায়েও ছিলেন অনেক কুশলী একজন ফুটবলার । যদিও তাঁর অধিনায়ক ('ক্যাপিতা') পরিচয়ের আড়ালে তা প্রায়ই ঢাকা পড়ে যায় । সর্বকালের অন্যতম সেরা এই ডিফেন্ডারকে জায়গা দিতে কাফু বাদ দিয়েছেন সর্বকালের অন্যতম সেরা রাইটব্যাককে! তিনি ১৯৫৪, ১৯৫৮ ও ১৯৬২ বিশ্বকাপে খেলা দালমা সান্তোস । লেফটব্যাক হিসেবে কাফু বেছে নিয়েছেন তাঁর দীর্ঘদিনের সতীর্থ রবার্তো কার্লোসকে । 'দ্য বুলেট ম্যান'কে বেছে নিতে গিয়ে কাফু জায়গা দিতে পারেননি ফুটবলীয় বুদ্ধিমত্তার জন্য পেলে-ভাভাদের কাছে 'এনসাইক্লোপিডিয়া' তকমা পাওয়া নিল্টন সান্তোসকে । কাফুর এই দুই সেন্টারব্যাক—আলদাইর আর লুসিও । এএস রোমার 'হল অব ফেম'-এ জায়গা পাওয়া আলদাইরের সঙ্গে এই ক্লাবে সতীর্থ ছিলেন কাফু । পরে তাঁকে ব্রাজিল দলেও পেয়েছেন

সতীর্থ হিসেবে । কাফুর ভাষ্য, আক্রমণ করতে গিয়ে মাঝেমধ্যে রক্ষণ ঠেকাতে থেই হারিয়ে ফেললে এই আলদাইরই বিপদমুক্ত করতেন তাঁকে, ব্রাজিলকে! কখনো বাজে কোনো ট্যাকল করতেন না, ছিল সঠিক সময়ে সঠিক জায়গায় থাকার দুর্দান্ত ক্ষমতা । অন্যদিকে লুসিও ছিলেন অপেক্ষাকৃত আক্রমণাত্মক ডিফেন্ডার । উচ্চতার জন্য বাতাসে যেমন দক্ষ ছিলেন, তেমনি বল পায়ে চকিত দৌড়ে হঠা করেই ঢুকে পড়তেন প্রতিপক্ষের রক্ষণভাগে । ড্রিবলিংয়েও পারঙ্গম 'দ্য হর্স' তকমা পাওয়া লুসিও এবং আলদাইর জুটি ব্রাজিলের হয়ে বেশ ভালো করত বলে কাফুর ধারণা । চারজন মিডফিল্ডারের মধ্যে ডিফেন্সিভ মিডফিল্ডার হিসেবে কাফু তাঁর একাদশে রেখেছেন ১৯৮২ বিশ্বকাপ দলের অন্যতম সেরা খেলোয়াড় ফ্যালকাওকে । কাফুর মতে, ফ্যালকাও কখনো ভুল পাস দিতেন না এবং সব সময়েই দলের ফরোয়ার্ডদের গোলে কোন না কোনোভাবে সহায়তা করতেন । তাঁকে রাখতে গিয়ে বেশ কজন কিংবদন্তি খেলোয়াড়কে বাদ দিতে হয়েছে কাফুকে—কথাটা তিনি নিজেই স্বীকার করেছেন । এর মধ্যে উল্লেখযোগ্য, ১৯৫৮ বিশ্বকাপে ব্রাজিলের মিডফিল্ড-মস্তিষ্ক জিতো, ১৯৯৪-এর বিশ্বকাপজয়ী অধিনায়ক দুঙ্গা এবং সত্তরের সেই দলের ডিফেন্সিভ মিডফিল্ডার ক্লদোয়ালদো । আক্রমণাত্মক মিডফিল্ডার ও উইঙ্গার হিসেবে কাফু তাঁর দলে জায়গা দিয়েছেন 'ইলাস্টিকো ড্রিবলিং কে নিখুঁত করে পরবর্তী প্রজন্মের কাছে তুলে ধরা রিভেলিনোকে । সঙ্গে তাঁর বহুদিনের সতীর্থ রিভালদো এবং 'সাদা পেলে' নামে খ্যাত জিকো । তাঁদের জায়গা দিতে কাফুর একাদশে স্থান হয়নি ৮২ বিশ্বকাপে ব্রাজিল দলে 'দার্শনিক'খ্যাত সক্রেটিস ও পেলের নায়ক—দিদির । বল নিয়ে রিভেলিনো আর জিকো আশ্চর্য সব কারিকুরি দেখাতে পারতেন । ইতিহাসের সেরা 'ড্রিবলার' হিসেবে এই রিভেলিনোকেই মানেন কাফু । ওদিকে রিভালদো আর ফ্যালকাওয়ের খেলার ধরন অনেকটা একই রকম ছিল । জিকো ছিলেন ফ্রি-কিকের জাদুকর । বক্সের আশপাশে ফাউল হলে জিকোর থেকে খুশি খুব সম্ভবত আর কেউই হতেন না! আর মাঠে রিভেলিনো, রিভালদো, জিকো আর ফ্যালকাও একসঙ্গে থাকলে ব্রাজিলকে বল হারানোর ভয় পেতে হতো না, কাফু এ ব্যাপারে নিশ্চিত ! ব্রাজিলের এই একাদশে কোচ হিসেবে কাফু নির্বাচন করেছেন ১৯৫৮ সালে খেলোয়াড় হিসেবে ও ১৯৭০ সালে কোচ হিসেবে বিশ্বকাপ জেতা মারিও জাগালোকে । আর হ্যাঁ, দলের দুই স্ট্রাইকার হিসেবে কাফুর পছন্দ পেলে আর রোনালদো । নিজ নিজ প্রজন্মে ব্রাজিল দলের এই দুই মূল খেলোয়াড় সমন্ধে কিছু বলার দরকার আছে কি ?

Sample Output 4

Statistical Approach:

আর মাঠে রিভেলিনো , রিভালদো , জিকো আর ফ্যালকাও একসঙ্গে থাকলে ব্রাজিলকে বল হারানোর ভয় পেতে হতো না , কাফু এ ব্যাপারে নিশ্চিত ! ১৯৯০ , ১৯৯৪ ও ১৯৯৮—তিনটি বিশ্বকাপ খেলা তাফারেলকে দলে রাখতে গিয়ে কাফু বাদ দিয়েছেন একসময়ের সতীর্থ দিদা , ১৯৫৮ ও ১৯৬২ সালের বিশ্বকাপজয়ী গোলরক্ষক জিলমার এবং ১৯৭০ বিশ্বকাপের এমারসন লিয়াওকে । ব্রাজিলের এই একাদশে কোচ হিসেবে কাফু নির্বাচন করেছেন ১৯৫৮ সালে খেলোয়াড় হিসেবে ও ১৯৭০ সালে কোচ হিসেবে বিশ্বকাপ জেতা মারিও জাগালোকে । বেছে নিয়েছেন , বিশ্বকাপের ইতিহাসে সর্বকালের অন্যতম সেরা দল হিসেবে খ্যাত '৭০-এর ব্রাজিল দলের অধিনায়ক—কার্লোস আলবার্তো তোরেসকে । সর্বকালের অন্যতম সেরা এই ডিফেন্ডারকে জায়গা দিতে কাফু বাদ দিয়েছেন সর্বকালের অন্যতম সেরা রাইটব্যাককে !

[Time: 0.455 Second]

Sentence Similarity Approach:

কাফুর এই দুই সেন্টারব্যাক—আলদাইর আর লুসিও । আর হ্যাঁ, দলের দুই স্ট্রাইকার হিসেবে কাফুর পছন্দ পেলে আর রোনালদো । খেলেছেন ব্রাজিলের ইতিহাসে সবচেয়ে বেশি ম্যাচ । 'সেলেসাও' গোলপোস্টের নিচে সবচেয়ে বেশি (১০১) ম্যাচ খেলা সাবেক

গোলরক্ষক । ১৯৯০, ১৯৯৪ ও ১৯৯৮—তিনটি বিশ্বকাপ খেলা তাফারেলকে দলে রাখতে গিয়ে কাফু বাদ দিয়েছেন একসময়ের সতীর্থ দিদা, ১৯৫৮ ও ১৯৬২ সালের বিশ্বকাপজয়ী গোলরক্ষক জিলমার এবং ১৯৭০ বিশ্বকাপের এমারসন লিয়াওকে ।

[Time: 100 Seconds]

A Random User:

পুরোনাম মার্কোস ইভানগেলিস্তা দি মোরাইস, অবশ্য ফুটবলবিশ্বে 'কাফু' নামেই জনপ্রিয়তা লাভ করা ব্রাজিলের কিংবদন্তী এই ফুটবলার সম্প্রতি ইংলিশ সংবাদমাধ্যম 'দ্য গার্ডিয়ান' কে ব্রাজিলের সর্বকালের সেরা একাদশ জানিয়েছেন। কাফুর মতে, ৪-৪-২ ছকে গড়া এই একাদশের সবাই অতি-আক্রমণাত্মক। ব্রাজিলের হয়ে সবচেয়ে বেশি (১০১) ম্যাচ খেলা সাবেক গোলরক্ষক ক্লদিও তাফারেলকে কাফু গোলপোষ্ট সামলানোর গুরুদায়িত্ব দিয়েছেন। রক্ষণ দেখাশোনার জন্য নিজেকে বিবেচনা না করলেও কাফু বেছে নিয়েছেন কার্লোস তোরেস, রবার্তো কার্লোস, আলদাইর এবং লুসিওকে। মিডফিল্ডের জন্য কাফু পছন্দ করেছেন ফ্যালকাও, রিভেলিনো, রিভালদো এবমগ জিকোকে। আক্রমণভাগে কাফুর নজর ছিলো রোনালদো এবং পেলের দিকে। পছন্দের একাদশ নির্বাচনে কাফুকে অনেক বেগ পেতে হলেও কোচ হিসেবে তিনি নির্বাচন করেছেন খেলোয়াড় এবং কোচ হিসেবে বিশ্বকাপ জেতা মারিও জাগালোকে।

[Time: 145 Seconds]

Analysis: If we talk about the output from the user, he tried to gather all the input in one short paragraph and it clearly denotes that someone is telling a story about Brazilian football. But comparing other two approach's output it is bit hard to get that someone is telling something. But the content of both approaches is similar and some lines are clearly matched. For the unclear details issue we can take the accuracy of both algorithms is 55%.

Sample Input 5 : 23 KB

সকালে উঠেই টপকোডারে এ লেখা দেখলাম "একটি শিশুকে একই আইফোন দিলে সে দিনরাত অ্যাংগ্রি বার্ডস খেলবে, শিশুটিকে কোডিং শিখালে সে আইফোনটার জন্য সফটওয়্যার তৈরি করবে" দারুণ এই লেখাটা দেখে মনে হলো কেন আমরা প্রোগ্রামিং বা কোডিং শিখবো সেটা নিয়ে বাংলায় কিছু লিখি ।

এ লেখাটি প্রোগ্রামিং নিয়ে যাদের কোনো ধারণা নেই বা খুব সামান্য ধারণা আছে তাদের আগ্রহী করে তোলার একটি ছোট্ট প্রচেষ্টা ।কম্পিউটার একটি অসম্ভব ক্ষমতাবান কিন্তু নির্বোধ একটি যন্ত্র । একটি যন্ত্র ৫০জন সাধারণ মানুষের কাজ একাই করতে পারে কিন্তু ৫০টি যন্ত্র একটি অসাধারণ মানুষের কাজ করতে পারেনা(Hubbard, Elbert) । প্রোগ্রামিং শিখে আমরা একেকজন হয়ে উঠতে পারি সেই মানুষটি যে এই যন্ত্রকে ইচ্ছামত কথা শোনাতে পারে । তুমি যা বলবে যেভাবে কম্পিউটার তাই করবে, এটাই হলো সোজা কথায় প্রোগ্রামিং । হয়তো বলতে পারো এখনইতো কম্পিউটার সেটা করে, আমি গান শুনাতে বললে সে শুনিয়ে দেয়, আমি গেম খেলতে চাইলে সে আমার সাথে খেলতে শুরু করে । কিন্তু আসল ব্যাপারটা হলো একজন প্রোগ্রামার আগেই কম্পিউটারকে বলে রেখেছে যে তুমি গান শুনতে চাইলে সে যেন শুনিয়ে দেয় । সে যদি বলে রাখতো গেম খেলতে চাইলে পড়তে বসার উপদেশ দিতে তাহলে কম্পিউটার তাই করতো, তোমার কিছু করার থাকতোনা । প্রোগ্রামার হলো সে যার কথায় কম্পিউটার উঠা-বসা করে ।

দারুণ একটা ব্যাপার এটা, তাইনা ?

কিন্তু তুমি কেন প্রোগ্রামিং শিখবে ? বড় বড় কথা বলার আগে সবথেকে প্রথম কারণ আমি বলবো কারণ "প্রোগ্রামিং দারুণ মজার একটি জিনিস" । কম্পিউটারের সাথে অন্য যন্ত্রের বড় পার্থক্য হলো এটা দিয়ে কতরকমের কাজ করানো যায় তার সীমা নেই বললে খুব একটা ভুল হবেনা । তাই প্রোগ্রামিং জানলে যে কতকিছু করা যায় তার তালিকা করতে বসলে শেষ করা কঠিন । তুমি দিনের পর দিন প্রোগ্রামিং করেও দেখবে জিনিসটা বোরিং হচ্ছেনা, প্রায় প্রতিদিনই নতুন মজার কিছু শিখছো, নতুন নতুন টেকনোলজী আবিষ্কারের সাথে সাথে তুমি আরো অনেক রকম কাজ করতে পারছো অথবা তুমিই করছো নতুন আবিষ্কার ! আজ হয়তো জটিল কোনো সমীকরণ সমাধান করার জন্য ফাংশন লিখছো, কাল এসব ভালো লাগছেনা বলে লাল-নীল রঙ দিয়ে একটি অ্যানিমেশন বানাতে বসে গেলে, তোমার সৃষ্টিশীলতার সবটুকুই কাজে লাগাতে পারবে প্রোগ্রামিং এর জগতে ।

একটি স্কুল-কলেজ পড়ুয়া ছেলেমেয়ে কম্পিউটার বা মোবাইল দিয়ে কি করে ? রাশিয়া-চীনের ছেলেমেয়েরা অনেকেই হয়তো অ্যাসেম্বলিতে কোড লিখে, কিন্তু জরিপ না করেও বলা যায় আমাদের দেশে বেশিভাগই মুভি দেখা, ফেসবুক , গেমস ছাড়া খুব বেশি কিছু করেনা । আসলে কম্পিউটার দিয়ে কি করা যায় তার ধারণাও অনেকের নাই । ছেলে বা মেয়েটিকে প্রোগ্রামিং শিখিয়ে দেয়া হলে তার জগ টাই পাল্টে যাবে । সে তখন সারাদিন গেমস না খেলে হয়তো একটি গেমস বানিয়ে ফেলবে । আমি বাংলাদেশেরই কিছু স্কুল-কলেজ পড়ুয়া প্রোগ্রামারদের জানি যারা বাংলা কিবোর্ড নিয়ে কাজ করে, ওপেন সোর্স কমিউনিটিতে অবদান রাখে । প্রোগ্রামিং জানলে তুমি বুঝতে পারবে কম্পিউটার শুধু বিনোদনের যন্ত্র নয়, কম্পিউটার তৈরা করা হয়েছিল এর ক্ষমতাকে ব্যবহার করে বড় বড় গবেষণা,হিসাব করার জন্য, তুমি যদি গবেষণা নাও করো অন্তত এই ক্ষমতাটা ব্যবহার শিখবে, সৃষ্টিশীল অনেক কাজ করতে পারবে । কম্পিউটারের জগতে অসাধারণ কিছু অগ্রগতি হয়েছে খুব কম বয়েসী প্রোগ্রামারদের দিয়ে, বিল গেটস স্কুলে থাকতেই চমকে দেয়ার মত কিছু প্রোগ্রাম লিখেছিলেন, প্রোগ্রামিং কনটেস্টে হাইরেটেড কোডারদের অনেকেই স্কুল-কলেজ এখনও শেষ করেনি ।

প্রোগ্রামিং করা মানে আনন্দের সাথে শেখা । এই শেখাটা খালি কম্পিউটারের মধ্য সীমাবদ্ধ না, অধিকাংশ ভালো প্রোগ্রামারদের খুবই ভালো গাণিতিক এবং লজিকাল জ্ঞান থাকে । দাবা খেলার মতোই প্রোগ্রামিং পুরোটাই লজিকের খেলা, কোন কাজের পর কোনটা করলে কি হবে, কিভাবে করলে আরো দ্রুত ফলাফল আসবে এইসব নিয়ে চিন্তা করতে করতে মস্তিষ্কের লজিকাল সেক্টরটা ডেভেলপ করে । আমার মতে চিন্তা করার মত আনন্দের এবং গুরুত্বপূর্ণ কাজ ২য়টি নেই । বিশেষ করে কম বয়সে প্রোগ্রামিং শিখালে সে চিন্তাশক্তি বৃদ্ধির যেই সুফলটা পাবে সেটা সারাজীবন কাজে লাগবে, সে যদি প্রোগ্রামিং পরে ছেড়েও দেয় তারপরেও চিন্তা করার ক্ষমতাটা থেকে যাবে ।

প্রোগ্রামিং কি শুধু কম্পিউটার সাইন্স যারা পড়ে বা পড়তে চায় তারা শিখবে ? সেটার কোনো যুক্তি নেই, তুমি যেই বিষয় নিয়েই পড়ছো বা পড়তে চাও, প্রোগ্রামিং তুমি আনন্দের জন্যই শিখতে পারো এবং চাইলে তোমার কাজেও লাগাতে পারো । তুমি বিজ্ঞানের যেকোনো বিষয়ে লেখাপড়া করলেতো কথাই নেই, তোমার গবেষণায় প্রতি মুহূর্তে কম্পিউটার লাগবে, তুমি বিজনেস, আর্টস পড়লেও প্রোগ্রামিং কাজে লাগবে । তুমি কোম্পানির জন্য দারুণ একটি ওয়েবসাইট বানাতে পারো, একটি সফটওয়্যার বানাতে পারো যেটা যেসব কাজ বোরিং সেগুলো স্বয়ংক্রিয় ভাবে করে দিবে ! আমি অনেক সময় ছোটো-খাটো কিন্তু বোরিং কাজ করার সময় চট করে একটা স্ক্রিপ্ট লিখে ফেলি, তারপর সেটাকে কাজ করতে দিয়ে ঘুম দেই !

প্রোগ্রামিং শেখা কি খুব কঠিন ? উত্তর হলো হ্যা,যদি তোমার আগ্রহ না থাকে এবং কেও তোমাকে জোর করে শেখায় । যদি একবার মজা পেয়ে যান তাহলে এরপর কারো শেখানো লাগবেনা, নিজেই সব শিখে ফেলতে পারো । আমার উপদেশ হবে ২-৩ সপ্তাহ প্রোগ্রামিং করার পর যদি তোমার ভালো না লাগে তাহলে জোর করে করার দরকার নাই, এটা তোমার জন্য না, অন্য যেটা ভালো লাগে সেই কাজ করো । যদি একবার ভালো লাগে বাজী ধরে বলতে পারি কোড লিখতে লিখতে তুমি প্রায়ই খাবার কথাও ভুলে

যাবে । যেকোন কাজের জন্যই সবথেকে গুরুত্বপূর্ণ ব্যাপার হলো ভালো লাগা, যেটা ভালো লাগেনা সেটা করার কোনো অর্থ আমি দেখিনা কারণ দুইদিন পর যা শিখসি সব ভুলে যাবো ।

শুরু কিভাবে করবে ? তোমার যদি ইন্টারনেট কানেকশন থাকে তাহলে কথাই নেই, ইন্টারনেটে অসংখ্য টিউটোরিয়াল আছে । ইংরেজীর পাশাপাশী বাংলা কিছু ভালো রিসোর্সও তুমি পাবে । যেমন শ্রদ্ধেয় রাগিব হাসানের shikkok.com ওয়েবসাইট বা ফাহিম ভাইয়ের পাইথন সাইট । এছাড়া খান একাডেমিতেও প্রোগ্রামিং এর ভিডিও আছে, বরাবরের মতই খুবই সুন্দর করে বুঝিয়েছেন সালমান খান । ইন্টারনেট না থাকলে তোমাকে বই জোগাড় করতে হবে, ব্যাক্তিগত ভাবে বিগিনারদের জন্য আমি ইন্টারনেটের থেকে বইকেই বেশি গুরুত্ব দিবো । প্রোগ্রামিং এর বইয়ের অভাব নেই দোকানে, তামিম শাহরিয়ার সুবিন ভাইয়ের একটি দারুণ বাংলা বই আছে । তবে একটা ব্যাপারে সতর্ক থাকবে ‘‘৭দিনে প্রোগ্রামিং শেখা’’ এই ধরণের চটকদার বইয়ের বা সাইটের ধারেকাছে যাবে, এগুলো সবকিছু ঝাপসা ভাবে শেখাবে, হার্ভার্ড শিল্ডের বইয়ের মত নামকরা এবং ভালো বই দেখে শিখো, বেসিক জিনিসগুলো পরিষ্কার হবে । এছাড়া লাগবে প্রোগ্রামিং এর জন্য কিছু সফটওয়্যার, এগুলোও সহজেই জোগাড় করতে পারবে । এরপর শুরু করে দাও কোড লেখা ! প্রথম ২ সপ্তাহ তোমার বেশ ঝামেলা লাগবে কারণ বিষয়টা নতুন, একটু পরপর আটকে যাবে, তারপর হঠা দেখবেন সবকিছু সহজ হয়ে গিয়েছে, মুহূর্তের মধ্যেই ১০০ লাইনের কোড লিখে ফেলেছো । প্রোগ্রামিং শেখার প্রধান শর্ত হলো হাল ছাড়া যাবেনা । প্রথম দিকে কোনো কোড কপি পেস্ট করবেনা, নিজের হাতে লিখবে ।

চাকরী-ক্যারিয়ার নিয়ে সবার মধ্যেই অনেক টেনশন থাকে । আনন্দের জন্য প্রোগ্রামিং শিখলেও এটা তোমার ক্যারিয়ারে খুবই গুরুত্বপূর্ণ । তুমি প্রোগ্রামিং জানলে নিশ্চিত থাকতে পারো কাজের কোনো অভাব জীবনে হবেনা । তুমি কোনো চাকরী না করেও ফ্রি-ল্যান্স কাজ করতে পারবে, এমনকি ছোটোখাট একটা কোম্পানিও খুলে বসতে পারবে । আমি আশেপাশে অনেককে দেখেছি কয়েক বন্ধু মিলে একটি ছোট কোম্পানি খুলে স্বাধীনভাবে কাজ করে, কি দারুণ একটা ব্যাপার ! প্রোগ্রামিং জানার আরেকটি দারুণ ব্যাপার হলো তুমি ভালো কোনো কাজ করলে খুব সহজেই সারা বিশ্ব জেনে যাবে । পৃথিবীর আরেক প্রান্তের মানুষ তোমার বানানো সফটওয়্যার দিয়ে গান শুনবে, তোমার অপারেটিং সিস্টেম বুট করবে, আবার পিসি হ্যাং করলে হয়তো তোমাকেই গালি দিবে ! গুগলের মতো কোম্পানিতে কাজ করতে চাইলে তোমার কিছু করতে হবেনা, আপনার কাজের খ্যাতিতে তারাই তোমাকে এসে অফার দিবে । তবে প্রোগ্রামিং শেখার উদ্দেশ্য কখনোই গুগলে চাকরী বা খ্যাতি অর্জন হওয়া উচিত নয়, শিখবে আনন্দের জন্য, জানার জন্য ।

সি বা জাভার মতো প্রোগ্রামিং ল্যাংগুয়েজ শেখা মানেই কিন্তু তুমি প্রোগ্রামিং শিখে ফেলোনি । ল্যাংগুয়েজ শেখা খুব সহজ কাজ, প্রথমে একটা কষ্ট করে শিখে ফেললে এরপর যেকোনো ল্যাংগুয়েজ শেখা যায় । তোমাকে খুবই ভালো লজিক ডেভেলপ করতে হবে, অ্যালগোরিদম আর ডাটা স্ট্রাকচার নিয়ে পড়ালেখা করতে হবে, গণিত জানতে হবে, তাহলেই তুমি একজন ভালো প্রোগ্রামার হয়ে উঠবে । তবে ভয়ের কিছু নেই, সবই তুমি ধীরে ধীরে শিখে ফেলতে পারবো, শুধু লাগবে চেষ্টা আর সময় । এটা আশা করবেনা যে ৬ মাসে তুমি অনেক ভালো প্রোগ্রামার হয়ে যাবে তবে লেগে থাকলে ২-৩ বছরে অবশ্যই মোটামুটি ভালো একটা লেভেলে তুমি পৌছাতে পারবে ।

তুমি যদি কম্পিউটার সাইন্সের স্টুডেন্ট হও তাহলে এইসব কথাই তুমি হয়তো জানো, শুধু বলবো প্রোগ্রামিং কে আর ৫টা সাবজেক্টের মতো ভেবোনা, খালি সিজিপিএ বাড়াতে কোডিং শিখলে তোমার মতো অভাগা কেও নাই, প্রোগ্রামিং উপভোগ করার চেষ্টা করো, জানার আনন্দে শিখো ।

আমার স্বপ্ন আমাদের দেশে একটা চিন্তা করার সংস্কৃতি তৈরি হবে । মানুষ একে অন্যের ব্যক্তিগত ব্যাপারে মাথা ঘামাবেনা, বরং

মাথা ঘামাবে গাণিতিক সমস্যা নিয়ে, পাজল নিয়ে, অ্যালগোরিদম নিয়ে । ছেলেমেয়েরা তাদের মেধা গেমস খেলার কাজে না লাগিয়ে কাজে লাগাবে পৃথিবীর উন্নয়নে । বই পড়া, গণিত চর্চা করার পাশাপাশি প্রোগ্রামিং শিখা এই সংস্কৃতি শুরু করতে বিশাল একটি ভুমিকা রাখতে পারে । আমি মনে করি বর্তমান যুগে প্রোগ্রামিং শেখাটা অন্য যেকোন বিষয় শেখার মতই গুরুত্বপূর্ণ, কারণ আমাদের সব কাজে কম্পিউটার লাগে । তাই আপনার আশেপাশের ছেলেমেয়েদের গেমস খেলতে দেখলে তাদের প্রোগ্রামিং সম্পর্কে জানাও, উ□সাহিত করো, অবশ্যই জোর করে শেখানোর কোনো মানে হয়না, যার ভালো লাগবে সে শিখবে তবে সবাই অন্তত জানুক প্রোগ্রামিং কি, এছাড়া কিভাবে শেখার জন্য উ□সাহিত হবে ? অনেকেই ইউনিভার্সিটিতে আসার আগে জানেনা প্রোগ্রামিং বলে একটা বস্তু আছে ! আর তোমরা প্রোগ্রামিং জানলে অন্যদেরও শিখতে সাহায্য করো, এভাবেই পরিবর্তন একসময় আসবেই, সবাই লজিক দিয়ে ভাবতে শিখবে, চিন্তা করার সংস্কৃতি তৈরি হবে ।

Sample Output 5

Statistical Approach:

তুমি যদি কম্পিউটার সাইন্সের স্টুডেন্ট হও তাহলে এইসব কথাই তুমি হয়তো জানো , শুধু বলবো প্রোগ্রামিং কে আর ৫টা সাবজেক্টের মতো ভেবোনা , খালি সিজিপিএ বাড়াতে কোডিং শিখলে তোমার মতো অভাগা কেও নাই , প্রোগ্রামিং উপভোগ করার চেষ্টা করো , জানার আনন্দে শিখো । আমার উপদেশ হবে ২-৩ সপ্তাহ প্রোগ্রামিং করার পর যদি তোমার ভালো না লাগে তাহলে জোর করে করার দরকার নাই , এটা তোমার জন্য না , অন্য যেটা ভালো লাগে সেই কাজ করো । আজ হয়তো জটিল কোনো সমীকরণ সমাধান করার জন্য ফাংশন লিখছো , কাল এসব ভালো লাগছেনা বলে লাল-নীল রঙ দিয়ে একটি অ্যানিমেশন বানাতে বসে গেলে , তোমার সৃষ্টিশীলতার সবটুকুই কাজে লাগাতে পারবে প্রোগ্রামিং এর জগতে । বিশেষ করে কম বয়সে প্রোগ্রামিং শিখালে সে চিন্তাশক্তি বৃদ্ধির যেই সুফলটা পাবে সেটা সারাজীবন কাজে লাগবে , সে যদি প্রোগ্রামিং পরে ছেড়েও দেয় তারপরেও চিন্তা করার ক্ষমতাটা থেকে যাবে । সি বা জাভার মতো প্রোগ্রামিং ল্যাংগুয়েজ শেখা মানেই কিন্তু তুমি প্রোগ্রামিং শিখে ফেলোনি । প্রোগ্রামিং জানার আরেকটি দারুণ ব্যাপার হলো তুমি ভালো কোনো কাজ করলে খুব সহজেই সারা বিশ্ব জেনে যাবে । তুমি বিজ্ঞানের যেকোনো বিষয়ে লেখাপড়া করলেতো কথাই নেই , তোমার গবেষণায় প্রতি মুহূর্তে কম্পিউটার লাগবে , তুমি বিজনেস , আর্টস পড়লেও প্রোগ্রামিং কাজে লাগবে । তবে একটা ব্যাপারে সতর্ক থাকবে " ৭দিনে প্রোগ্রামিং শেখা " এই ধরণের চটকদার বইয়ের বা সাইটের ধারেকাছে যাবে , এগুলো সবকিছু ঝাপসা ভাবে শেখাবে , হার্ভার্ড শিল্ডের বইয়ের মত নামকরা এবং ভালো বই দেখে শিখো , বেসিক জিনিসগুলো পরিষ্কার হবে । প্রোগ্রামিং জানলে তুমি বুঝতে পারবে কম্পিউটার শুধু বিনোদনের যন্ত্র নয় , কম্পিউটার তৈরা করা হয়েছিল এর ক্ষমতাকে ব্যবহার করে বড় বড় গবেষণা , হিসাব করার জন্য , তুমি যদি গবেষণা নাও করো অন্তত এই ক্ষমতাটা ব্যবহার শিখবে , সৃষ্টিশীল অনেক কাজ করতে পারবে । সেটার কোনো যুক্তি নেই , তুমি যেই বিষয় নিয়েই পড়ছো বা পড়তে চাও , প্রোগ্রামিং তুমি আনন্দের জন্যই শিখতে পারো এবং চাইলে তোমার কাজেও লাগাতে পারো ।

[Time: 0.16 Seconds]

Sentence Similarity Approach:

আমার স্বপ্ন আমাদের দেশে একটা চিন্তা করার সংস্কৃতি তৈরি হবে । আর তোমরা প্রোগ্রামিং জানলে অন্যদেরও শিখতে সাহায্য করো, এভাবেই পরিবর্তন একসময় আসবেই, সবাই লজিক দিয়ে ভাবতে শিখবে, চিন্তা করার সংস্কৃতি তৈরি হবে । প্রোগ্রামিং করা মানে আনন্দের সাথে শেখা । আনন্দের জন্য প্রোগ্রামিং শিখলেও এটা তোমার ক্যারিয়ারে খুবই গুরুত্বপূর্ণ । তুমি যা বলবে যেভাবে কম্পিউটার তাই করবে, এটাই হলো সোজা কথায় প্রোগ্রামিং । প্রোগ্রামার হলো সে যার কথায় কম্পিউটার উঠা-বসা করে । আমার

মতে চিন্তা করার মত আনন্দের এবং গুরুত্বপূর্ণ কাজ ২য়টি নেই । দারুণ একটা ব্যাপার এটা, তাইনা ? আমি আশেপাশে অনেককে দেখেছি কয়েক বন্ধু মিলে একটি ছোট কোম্পানি খুলে স্বাধীনভাবে কাজ করে, কি দারুণ একটা ব্যাপার ! তাই প্রোগ্রামিং জানলে যে কতকিছু করা যায় তার তালিকা করতে বসলে শেষ করা কঠিন ।

[Time: 325 Seconds]

A Random User:

প্রোগ্রামিং অনেক মজার জিনিস । প্রোগ্রামিং এর মাধ্যমে কম্পিউটার নামক বোকা যন্ত্রকে দিয়ে অনেক কঠিনতর কাজ সহজে করিয়ে নেয়া যায় । প্রোগ্রামিং মানুষকে চিন্তা করতে শেখায় । সৃজনশীলতার মাধ্যমে নতুন কিছু তৈরি করতে শেখায় । তবে কেবল মাত্র অর্থোপার্জন কিংবা খ্যাতির জন্য প্রোগ্রামিং শিখে বেশিদূর যাওয়া যায় না । প্রোগ্রামিং শিখতে গেলে অদম্য ধৈর্য্য আর কঠোর চেষ্টা প্রয়োজন । তবেই সাফল্য আসতে বাধ্য ।

[Time: 255 seconds]

Analysis: Comparing both algorith's output it is clear that both are talking about the importance of programming which also the main theme of the actual content. And the random user also illustrates the same thing. In the sentence similarity approach the algorithm succeded to figure out the hope of the author. But in the statistical approach it can not. Based on this issue we can set the accuracy of 50%.

4.2 Discussion

While processing the randomized inputs we have seen that most of the Statistical Approach Summarizer returns the lengthy lines which contain more information and also in a very short time. Also as there is no use of complex algorithms and techniques and hence processing time is pretty lower for Statistical Approach. Whenever we read about the output of the statisctical approach we have seen that it somehow miss some important gist. Where as sentence similarity approach mostly do not miss the gists. The sentence similarity takes much more time with respect to the statistical approach but it returns the less sized line mostly. And also it gives a very realistic result.

One thing that we have noticed is with increasing size of text the run time is increasing almost exponenetial way for sentence similarity approach where as the statistical approach do not have such characteristics. The complete timing issue can be shown as the following graph –

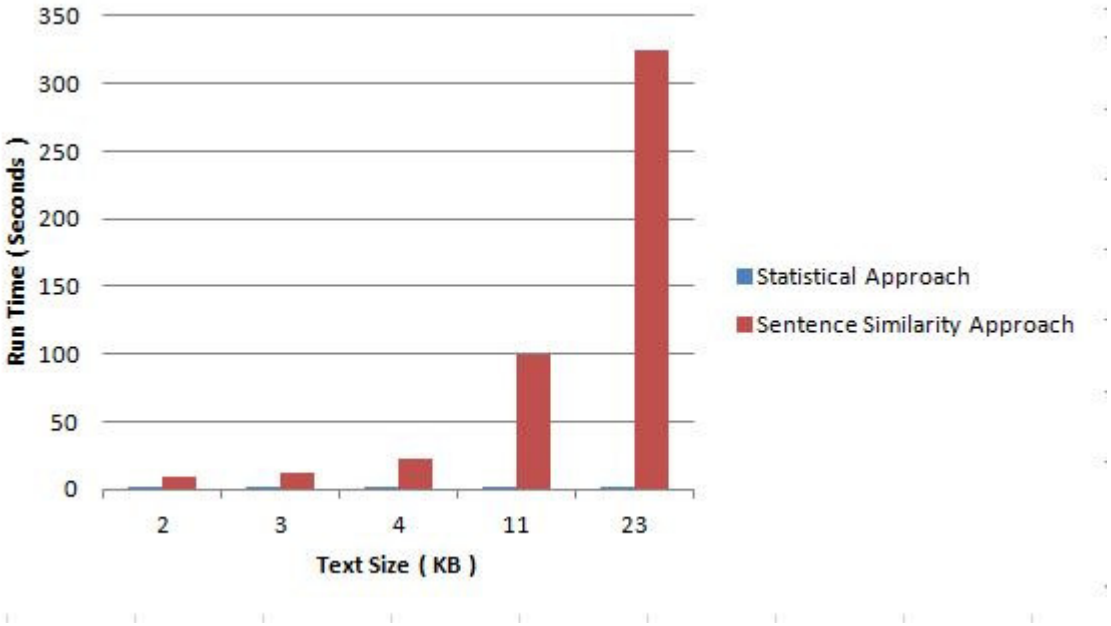

Figure 4: Text size / run time analysis

Above analysis show the text size – run time analysis bar chart. The slightly blue shades on the graph denote run time of Statistical Approach. And the red bar denotes the run time for Sentence Similarity approach.

4.3 Accuracy

From the analysis for each of the inputs we are able to see that both of our approaches at least capable to show the gists of the main content. Reading the output we can at least gather an idea that what it is talking about. In statistical analysis we missed some of the gists for Sample Input 5. Where as sentence similarity managed to show that gists. From this point of view, sentence similarity is more accurate. Although time complexity of sentence similarity is a big deal where as statistical analysis takes much less time. Apart from that we think that somehow all of our idea managed to see the daylight.

Chapter 5

Chapter 5 : Conclusion and Future Works

Through out the whole project we have attempted to implement the Statistical approach and Sentence similarity approach to summarize a single text docomument in Bangla language. We have analyzed the samples and calculated the numbers. From the analysis we have come to decision that to make things more efficient and accurate we need to design our own Bangla WordNet and corpus. And then only it might be able to get better output.

5.1 Conclusion

i. The senetence similarity based summarizer is able to give a satisfactory result for a wide range of documents.
ii. The statistical approach is able to process a large amount data at a very short time.
iii. The sentence similarity approach provides more realistic result and contains more gists than statistical approach.
iv. Lack of Bangla WordNet and Corpus decrease the accuracy of Sentence similarity approach.
v. Caching of the calculation may also be a great choice to reduce the run time of Sentence similarity.

5.2 Future Works

i. Apply caching technique of information in sentence similarity approach to make things faster.
ii. Build a good WordNet and Corpus to get more accurate result.
iii. Build an official Bangla Summarizer module for python.
iv. Create a GUI for summarization.

References

[1] J. Allen, Natural Language Understanding. Redwood City, Calif.: Benjamin Cummings, 1995.

[2] J. Atkinson-Abutridy, C. Mellish, and S. Aitken, “Combining Information Extraction with Genetic Algorithms for Text Mining,” IEEE Intelligent Systems, vol. 19, no. 3, 2004.

[3] Brown Corpus Information, http://clwww.essex.ac.uk/w3c/corpus_ling/content/corpora/list/private/brown/brown.html, 2005.

[4] A. Budanitsky and G. Hirst, “Semantic Distance in WordNet: An Experimental, Application-Oriented Evaluation of Five Measures,” Proc. Workshop WordNet and Other Lexical Resources, Second Meeting of the North Am. Chapter of the Assoc. for Computational Linguistics, 2001.

[5].C. Burgess, K. Livesay, and K. Lund, “Explorations in Context Space: Words, Sentences, Discourse,” Discourse Processes, vol. 25, nos. 2-3, pp. 211-257, 1998.

[6] W.G. Charles, “Contextual Correlates of Meaning,” Applied Psycholinguistics, vol. 21, no. 4, pp. 505-524, 2000.

[7] J.H. Chiang and H.C. Yu, “Literature Extraction of Protein Functions Using Sentence Pattern Mining,” IEEE Trans. Knowledge and Data Eng., vol. 17, no. 8, pp. 1088-1098, Aug. 2005.

[8] T.A.S. Coelho, P.P. Calado, L.V. Souza, B. Ribeiro-Neto, and R. Muntz, “Image Retrieval Using Multiple Evidence Ranking,” IEEE Trans. Knowledge and Data Eng., vol. 16, no. 4, pp. 408-417, Apr. 2004.

[9] G. Erkan and D.R. Radev, “LexRank: Graph-Based Lexical Centrality As Salience in Text Summarization,” J. Artificial Intelligence Research, vol. 22, pp. 457-479, 2004.

[10] P.W. Foltz, W. Kintsch, and T.K. Landauer, “The Measurement of Textual Coherence with Latent Semantic Analysis,” Discourse Processes, vol. 25, nos. 2-3, pp. 285-307, 1998.

[11] P. Wiemer-Hastings, “Adding Syntactic Information to LSA,” Proc. 22nd Ann. Conf. Cognitive Science Soc., pp. 989-993, 2000.

[12] V. Hatzivassiloglou, J. Klavans, and E. Eskin, “Detecting Text Similarity over Short Passages: Exploring Linguistic Feature Combinations via Machine Learning,” Proc. Joint SIGDAT Conf. Empirical Methods in NLP and Very Large Corpora, 1999.

[13] V. Hatzivassiloglou, J. Klavans, and E. Eskin, “Detecting Similarity by Applying Leaning over Indicators,” Proc. 37th Ann. Meeting of the Assoc. for Computational Linguistics, 1999.

[14] D. Jurafsky and J.H. Martin, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall, 2000.

[15] Y. Ko, J. Park, and J. Seo, “Improving Text Categorization Using the Importance of Sentences,” Information Processing and Management, vol. 40, pp. 65-79, 2004.

[16] H. Kozima, “Computing Lexical Cohesion as a Tool for Text Analysis,” PhD thesis, Course in Computer Science and Information Math., Graduate School of Electro-Comm., Univ. of ElectroCommunications, 1994.

[17] T.K. Landauer, D. Laham, B. Rehder, and M.E. Schreiner, “How Well Can Passage Meaning Be Derived without Using Word Order? A Comparison of Latent Semantic Analysis and Humans,” Proc. 19th Ann. Meeting of the Cognitive Science Soc., pp. 412-417, 1997.

[18] T.K. Landauer, P.W. Foltz, and D. Laham, “Introduction to Latent Semantic Analysis,” Discourse Processes, vol. 25, nos. 2-3, pp. 259- 284, 1998.

[19] T.K. Landauer, D. Laham, and P.W. Foltz, “Learning Human-Like Knowledge by Sngular Value Decomposition: A Progress Report,” Advances in Neural Information Processing Systems 10, M.I. Jordan, M.J. Kearns, and S.A. Solla, eds., Cambridge, Mass.: MIT Press, pp. 45-51, 1998.

[20] Y.H. Li, Z. Bandar, and D. McLean, “An Approach for Measuring Semantic Similarity Using Multiple Information Sources,” IEEE Trans. Knowledge and Data Eng., vol. 15, no. 4, pp. 871-882, July/ Aug. 2003. [21] Y. Liu and C.Q. Zong, “Example-Based Chinese-English MT,” Proc. 2004 IEEE Int’l Conf. Systems, Man, and Cybernetics, vols. 1-7, pp. 6093-6096, 2004.

[22] J.L. McClelland and A.H. Kawamoto, “Mechanisms of Sentence Processing: Assigning Roles to Constituents of Sentences,” Parallel Distributed Process 2, D.E. Rumelhart, J.L. McClelland, and the PDP Research, eds., pp. 272-325, MIT Press, 1986.

[23] M. McHale, “A Comparison of WordNet and Roget’s Taxonomy for Measuring Semantic Similarity,” Proc. COLING/ACL Workshop Usage of WordNet in Natural Language Processing Systems, 1998.

[24] C.T. Meadow, B.R. Boyce, and D.H. Kraft, Text Information Retrieval Systems, second ed. Academic Press, 2000.

[25] D. Michie, “Return of the Imitation Game,”

[26] Verloren, Wikipedia Data Mining Article, 2002.

[27] Contributor with an User IP 217.121.150.77 , Wikipedia Text Mining Article, 2003

[28] Juan-Manuel Torres-Moreno, Automatic Text Summarization (Cognitive Science and Knowledge Management) 1st Edition , 2014.

[29] Inderjeet Mani, Mark T. Maybury, Advances in Automatic Text Summarization (The MIT Press), 1999.

[30] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, Krys Kochut, Text Summarization Techniques: A Brief Survey, 2017.

[31] Yogan Jaya Kumar, Ong Sing Goh, Halizah Basiron, Ngo Hea Choon , Puspalata C Suppiah, A Review on Automatic Text Summarization Approaches, 2016.

[32] Yazan Alaya AL-Khassawneh, Naomie Salim and Mutasem Jarrah, Improving Triangle-Graph Based Text Summarization using Hybrid Similarity Function, 2017.

[33] Dwijen Rudrapal, Amitava Das, Baby Bhattacharya, Measuring Semantic Similarity for Bengali Tweets Using WordNet, 2015

[34] Manjira Sinha, Abhik Jana, Tithankar Dasgupta, Anupam Basu, A semantic lexicon and similarity measure in Bangla, 2013.