International Journal of Multimedia and Ubiquitous Engineering
Vol.10, No.5 (2015), pp.263-276
http://dx.doi.org/10.14257/ijmue.2015.10.5.24

A Graph Theoretical Preprocessing Step for Text Compression

Kaushik K. Phukon[1] and Hemanta K. Baruah[2]

[1]*Department of Computer Science, Gauhati University,*
[2]*Vice-Chancellor, Bodoland University*
Assam, India.
[1]*kaushikphukon@gmail.com,*[2]*hemanta_ bh@yahoo.com*

Abstract

This paper presents $CSGM_2$, a text preprocessing technique for compression purposes. It converts the original text into a word net (graph representation) and can retain the detailed contextual information such as word proximity. Specific directed graph is proposed to model this word net where words are stored in vertices and edges represent word transitions. The word net is fully capable of holding the natural word order in the original text and hence can be used directly for encoding purposes.

Keywords: *Preprocessing, Graph, Text, Compression*

1. Introduction

Data compression has immense importance in the areas of data storage and data transmission despite of the high speed data networks and large capacity storage devices are available these days. The general theory of data compression techniques on text files is to transform a string of characters into a new string which contains the same information but with new length as small as possible [1].

Thus, compressed data occupies less storage space and takes a lesser amount of time to be read from disk or transmitted over a communication link [2]. The cost of data compression is basically the time needed to encode and decode the data. However, now a days this is becoming less significant as the speed of processor is increasing much faster than data transmission or disk transfer speeds.

Natural language texts have information meant for general-purpose person to person conversation. Hence, its structure follows the syntactic rules used to create these conversations. We generally do not consider natural language text as character sequences. We consider them as a sequence of groups of alphabetic symbols forming words. Hence the character-based compression techniques are barely able to capture and use long-range correlations existing between words. However, if text compression techniques were to use bigger units than single characters as the basic storage element, they would be able to capture and use long-range correlations [3]. Thus, a word-based algorithm is a batter choice when a natural text is compressed as words are the true reflector of text entropy [4].

The goal of Text compression is to identify and eliminate redundant information to achieve a more compact text representation. Statistical and dictionary-based techniques are generally applied for text compression. However there is a third category of approach based on the combination of text preprocessing and universal compression. In this category of approach, also known as preprocessing based compression, the preprocessing algorithms are performed before the actual compression process. Preprocessing routines are not compression algorithms, but they output less redundant text representations. The well formatted text representations are then compressed with common techniques. Thus, preprocessing-based compressors can be segregated into two independent parts: preprocessing and standard compression algorithms.

ISSN: 1975-0080 IJMUE

The $CSGM_2$ technique, explained in this paper, was originally defined as CSGM (Context Sensitive Graph Model), a graph model for representing large text portion of web documents[5]. It transforms the original text into a word net by considering that the information contained in the text does not only depend on single words but also on the different relationships between them. For example, let us focus our attention on a set of nine words (‘only’, ‘he’, ‘told’, ‘me’, ‘that’, ‘he’, ‘will’, ‘accompany’, ‘us’) used in a text. If we find a phrase such as “...*ONLY he told me that he will accompany us*...”, we perceive that he is the only one. But if the phrase is “...*He ONLY told me that he will accompany us*...”, we perceive that he told ‘me’ nothing more. Again if the phase is “...*He told ONLY me that he will accompany us*...”, we perceive that he told no one else. Although these phrases transmit different information, they are similarly represented if correlation between them is not taken into account. Thus, $CSGM_2$ regards a text as a sequence of transitions between adjoining words and outputs an intermediate graphical representation which can be obviously compressed with universal techniques. The original work [5] shows how CSGM effectively performs in conjunction with web documents.

2. Related Work

So many known methods of data compression are there. Although the compression techniques are based on different ideas, are suitable for different types of data, and produce different results, but they are all based on the same principle. They compress data by removing redundancy from the source file. Any nonrandom collection of data must have some structure, and this structure can be exploited to achieve a smaller representation of the data where no structure is discernible. Thus, redundancy significantly matters in any discussion related to data compression [4].

Statistical and dictionary-based techniques are traditionally considered for text compression. Both the methods are briefly stated below together with a third approach based on preprocessing.

2.1 Statistical Compression

In general, statistical compression techniques apply two complementary phases of modeling and coding. The modeling phase obtains a probability distribution of the concerned text, which is then used as the basis for codification. The probability of each symbol is calculated on the basis of global text statistics or depends on a context of n preceding symbols, where n is a user defined parameter. After the first phase the coding phase assigns code-words to each symbol following the probability distribution obtained in the first phase and uses them to get the compressed text representation.

Finding the Probability Distribution of Text. At first, this phase requires a definition of a symbol whether it is considered to be a character, a fixed number of characters or a word, defined as a maximal sequence of alphanumeric/nonalphanumeric characters. The set of all different symbols in the text is known as alphabet. In natural language text compression we generally have two alternatives: word and character based alphabet. The compression effectiveness is subject to the alphabet choice. For example, in natural language texts, more skewness is observed in a word-based distribution than a character-based one [6]. Hence, better effectiveness can be achieved if a word alphabet is used rather than a character one.

Structured, Semi structured and unstructured methods can be applied to implement the first phase. The decession is influencel to the next phase because each of the method has its own set of operational procedures to follow. In structured techniques the coder and the decoder shares in advance an expected statistical distribution. This technique does not work well when there is a significant deviation from the expected standard. Instead of following a predetermined or expected statistical distribution, the Semi-structured

techniques learn it during a first pass. In accordance with the knowledge acquired in the first pass the required statistic to model the text is determined in the second pass. The unstructured method also learns about the text or data along with the run, but in this case only one run is there. The method must perform the essentialities of the modeling task for the given text or data in a single pass only. Unstructured compression method usually works without any external information which makes such type of compression methods more general then that of other methods [4].

Coding. In this phase, based on the statistic calculated in the first phase the text symbols are assigned specific code words according to some criteria. The symbols with relatively high frequency are encoded with short code-words, whereas the symbols occurring with relatively low frequency are encoded with longer code-words. This conversion is performed to reduce the average lengths of the code words to some standard.

The Huffman algorithm [7] is one of the best known compression algorithm in general. It is a method supporting minimum redundancy in which symbols are assigned variable-bit code-words. Here also the text symbols with relatively high frequency are encoded with shortest code-words and vice versa.

Two different strategies can be adopted to compute the statistical distribution of the text symbols [4]. In one strategy the statistical distribution of the text symbols is calculated based on the entire text *i.e.* based on the total number of symbols and the frequency of a particular symbol appearing within the text. This type of model is known as 0 order model. On the other hand, in k order models the distribution is determined on the basis of some prefixed number of preceding symbols. Both the strategies have their own characteristics based on different parameters.

2.2. Dictionary-based Compression

In A dictionary-based compression method the input text symbols are replaced by bit strings. In this method we have a dictionary of words and the corresponding bit strings. The dictionary plays the role of a mapping function in between the original text and the file containing bit strings. The dictionary method converts the original file containing text symbols into bit strings, and thereby reducing the text size according to the performed word-codeword mapping. As opposed to the statistical compression, in dictionary based compression, we do not experience two different phases of modeling and coding. Moreover no statistical probability distribution is assigned to the textual symbols [8].

These methods also comprise two relatively independent phases. In the first phase, a dictionary is constructed and unique bit strings are assigned to various symbols comprising the text. In the second phase a compressed text representation is obtained by compressing the resultant text file obtained by substituting the text symbols with bit strings.

LZ77 [9] and LZ78 [10], are the well-known dictionary-based compressors. These methods implement an adaptive policy to construct their dictionary. LZ is a well known family of compression algorithms as these algorithms are capable of deriving more compressed text file using limited resources such as memory and also with reasonably good speed of compression and decompression. Different popular compressors are designed on the LZ platform such as gzip and p7zip [12].

2.3. Preprocessing-based Compression

With a view to attain high compression ratio, Preprocessing based compression algorithms are proposed comprising a preprocessing step significantly different than that of compression algorithms. This preprocessing step generally converts the original text into a different but reversible form with lesser amount of space, reducing redundancy, if any. The Preprocessing based compression algorithms consist of two independent parts,

first one being the preprocessing procedures. The output from the first phase is used as an input to the second part containing compression procedures. Different commercial preprocessing based compression software are available in the market among which bzip2 is one of the popular compressor. This compressor is based on the Burrows-Wheeler Transform (BWT) [11] which is one of the best known preprocessing algorithm for compression.

3. Natural Language Text Modeling and Compression

Natural language texts contain information used by humans for general-purpose communication. Although the text intended for general purpose communication is extremely rich and varied, it is also predictable. This is mainly based on the fact that all natural languages have a predefined set of words and are the basic units used in human communications. A natural language text file consists of an alternation of words and separators (*e.g.* punctuation, space or other spatial characters). It was found that about 70%-80% of the separators are single spaces [12].

In natural language texts the word distributions are much skewed: there are few words that have extreme frequencies and many words which have low frequencies. This feature is approximated in Zipf 's Law [12]. Zipf's Law attempts to capture the distribution of word frequencies in the text. That is, the relative frequency of the i^{th} most frequent word is $1/i^{\theta}$, for some $1 < \theta < 2$ (in English texts). On the other hand, the number of distinct words in a document (referred to as vocabulary) is an important feature in natural language modeling. Heaps' law [6] helps to predict the growth of the vocabulary size in natural language texts. It establishes that the number of different words in a text of n words is $O(n^{\beta})$, for some β between 0.4 and 0.6. Thus, the vocabulary grows sublinearly with the collection size.

Based on Word and separator distributions in natural language it is possible to find some ways to model texts. One such possibility is to An apparent possibility is to consider the different inter-word separators as symbols and make a single alphabet for words and separators. This idea does not support the fundamental alternation property stated above [13]. Another idea is to use two different alphabets: one for words and one for separators. Once it is confirmed that the text begins with a word or a separator, we can proceed with the idea without any confusion about which alphabet to use. This model is called separate alphabets. As stated above, this model is heavily influenced by the high frequency of single spaces distribution. As previously cited, about $70 - 80\%$ of separators are single spaces. Therefore we can consider the single spaces as default separators, which is the idea behind the spaceless words transformation. According to this idea, if a word is followed by a single space, only the word will be encoded. If not, both the word and the separator will be encoded. Here the alternation property does not hold any more. The single space is excluded and we have a single alphabet for words and separators. This spaceless word transformation achieves better compression ratios and can be considered as the de facto standard in word-based modeling for English text [14].

Some traditional compressors like Huffman have been reengineered to perform on word vocabularies. This new form of Huffman algorithm is a popular word based compression algorithm. It retains the original Huffman algorithm features, but process the text as a sequence of words and replaces the original words with variable-bit code-words to represent them. This word-based Huffman largely outperforms the traditional character-based one by achieving near about 25% compression ratio instead of 65%, achieved by the character-based Huffman [12].

3.1. Graph-based Modeling for Natural Language Texts

The Graph-theoretical objects can be treated as a natural way of modeling one's impressions of a text. As the graphical objects are able to preserve the connections or relationship between its constituents of varying granularity [15]. With the help of the two fundamental components of graph structures namely vertices and edges, we can represent a text, hardly losing any information. Depending on the application at hand, text units of various sizes and characteristics like a single word, a word sequence or others. Graph theory is the only such mathematical tool that dictates the type of relations used to draw connections between any two vertices.

In different sub areas of natural Language processing, the graph-based modeling is widely used. These approaches represent the text by means of a graph, in which words or other text entities with meaningful relations are interconnected through vertices and edges. The information provided by the word order in a text is enough to construct a connected directed graph [15].

4. The $CSGM_2$ Transformation

4.1. Basic Concepts of Graph Theory

This section covers a brief revision of the basic concepts of graph theory [16].

Definition 1 (Graph). A graph G is a 4-tuple: $G= (V, E,\alpha,\beta)$, where V is a set of nodes (vertices), $E \subseteq V \times V$ is a set of edges connecting the nodes, $\alpha : V \rightarrow \Sigma v$ is a function labeling the nodes, and $\beta : V \times V \rightarrow \Sigma e$ is a function labeling the edges (Σv and Σe being the sets of labels that can appear on the nodes and edges, respectively). For brevity, we may refer to G as $G= (V, E)$ by omitting the labeling functions.

Definition 2 (Ordered Pair). An ordered pair is a collection of two objects such that one can be distinguished as the first element and the other as the second element. An ordered pair with first element x and second element y is written as (x, y).

Definition 3 (Digraph). Let V be a set of vertices and E a set of ordered pairs, i.e. a subset of $V \times V$, called arcs or directed edges. Then, a directed graph or digraph G is an ordered pair $G = (V, E)$ where V is the set that contains all the vertices of G and $E \subseteq V \times V$ is the set which groups all the edges form G. Therefore, ordering is significant in an ordered pair and consequently a pair of objects (x, y) is considered distinct from (y, x), for $x \neq y$. In graph theory, the first vertex in a directed edge is called the source and the second vertex is called the destination. In this paper, we shall refer to a directed edge as edge.

Definition 4 (Degree). The degree of a vertex in a graph is the number of edges connected to it. If the graph is a directed graph, the in-degree of $v \in V$ is the number of edges where v is the destination and the out-degree of $v \in V$ is the number of edges where v is the source, so the degree is the sum of the in-degree with the out-degree. Hence the in-degree of $v \in V$ is defined by $\deg^+(v) = |D^+(v)|$ and the out-degree of $v \in V$ is defined by $\deg^-(v) = |D^-(v)|$.

4.2. Word Net Building

The $CSGM_2$ technique transforms the original text into a word net (graph) in which all relationships between adjoining words are retained. For all kinds of natural language natural, all words in a sentence have some relationship to all other words in it. The closer two words are to each other; the stronger their connection tends to be [12] and graphs are the right scientific structures to represent such relationship. In the rest of the paper we are

going to present how the $CSGM_2$ technique performs on a text T of length n words, drawn from a vocabulary $\sum$ containing σ different words.

To implement the word net A digraph G := (V, E) is considered in accordance with the following features:

• Each unique term appearing in the document becomes a node in the graph representing that document. Each node is labeled with the term it represents and a global identifier, which is a non negative integer value $x \in [1, \sigma]$ where σ is the total word count in T. The node labels in a document graph are unique, since a single node is created for each keyword even if a term appears more than once in the text.

• Second, if word *a* immediately precedes word *b* somewhere in the document, then there is a directed edge from the node corresponding to term *a* to the node corresponding to term *b* with an edge labeled as (n:b), where n is a user specified parameter representing the distance between the two words and b is the is the number of times that the edge has been traversed.

In this model , instead of considering only terms immediately following a given term in a document, we look up to *n* terms (here we are considering n=2) ahead and connect the succeeding terms with an edge labeled as (n:b).

• There are no multiple edges in G. If there is more than one transition between two consecutive words ω_x and ω_y (represented in x and y vertices respectively) only a single edge, (x, y), is modeled.

This representation can be used in conjunction with all modeling ways considered to symbolize natural language texts. Even so, how separators are modeled is an important issue which directly affects the space/time tradeoff of the $CSGM_2$ technique.

Single Alphabet. In this technique a unique graph is used to model T over a single alphabet, which comprises words and separators. This modeling policy does not support the alternation property between words and separators. This choice is dominated by the vertex which represents the single space. It overloads the model as it has higher out-degrees. In addition, two adjoining words are never connected by considering that these are related with separators. Thus, a $CSGM_2$ modeling, based on the single alphabet, appears to be poor in text representation because it is not able to directly identify correlations between words.

Separate Alphabets. These modeling technique two different alphabets are used for words and separators to deal the alternation property. Two independent graphs are used to implement the idea for each class of symbols, taking into consideration the word and separator alternation. Thus, T is divided into two parts of the same size that are represented in each graph. On one part, correlations between words are identified and modeled achieving a good text representation. On the other part, the separator representation is totally dominated by the single space symbol. A very frequent edge describes a self-loop in the vertex in which it is represented. This fact will affect $CSGM_2$ effectiveness due to inclusion of dispensable information.

Spaceless Words. This is based on a single alphabet in which the single space is not counted separately. Thus a single graph is required to model T. Spaceless words is able to take advantage of the positive features of both the previous modeling methods. In addition, it is not influenced by the single space effect. Thus, a rich text representation is

achieved in which word relationships are also identified and modeled in order to improve the $CSGM_2$ coding task.

We choose spaceless words to model T on $CSGM_2$. This decision is made by considering the experimental outcomes of other researchers, which report that the spaceless words choice allows more compact word net representations to be achieved with respect to separate alphabets (its size is reduced by between 30% and 40%). Considering this fact, the term *word* is used to refer to both text words and separators in the discussions below.

4.3 Transforming Natural Language Text through a Word Net

A word net, implemented through a graph G := (V, E), is adaptively built as T is processed. Thus, graph and text representations are coordinated at the same time as the word net is incrementally built and updated. This can be illustrated with a simple example. We keep $c \in V$ as the current vertex (c stores the word say $\omega_c \in \sum$) and suppose that the word ω is reached. The three following situations can happen:

• If ω is reached for the first time a vertex v is appended to V for storing this new word, and a new edge (c, v), from the current vertex, is inserted in E. This edge represents the word transition followed from ω_c to ω.

• If ω is previously appeared (it is stored in $v \in V$), but it was never preceded by ω_c. Then a is new edge (c, v) must be appended to E to represent this new word transition.

• If ω previously appeared and, at some time, was preceded by the word represented in c then it implies that the edge $(c, v) \in E$, and it is locally identified in c with a non-negative value x. The statistical information of (c, v) is updated.

The following algorithm formalizes the $CSGM_2$ transformation process.

Algorithm 1

1: V $\leftarrow \phi$
2: E $\leftarrow \phi$
3: current $\leftarrow$ StartVertex;
4: previous $\leftarrow$ null
5: while there are more words in T do
6: word $\leftarrow$ T.ParseWord();
7: if word $\notin$ V then
8: V $\leftarrow$ V $\bigcup$ word;
9: destination $\leftarrow$ V.retrieveID(word);
10: If current $\neq$ destination
11: edge $\leftarrow$ (current, destination);
12 If prev $\neq$ current
13 edge $\leftarrow$ (prev,destination)
14 else goto step 17
15 end if
16: E $\leftarrow$ E $\bigcup$ edge;
17: G_{text}.encode(V);
18: G_{vertex}.encode(word);

```
19:                 G_edge.encode(edge)
20:             else previous ← current
21:                  current ← destination;
22:             end if
23:         else
24:                 destination ← V.retrieveID(word);
25:                 edge ←  (current, destination);
26:                 if  edge ∉ E then
27:                     E ← E ∪ edge;
28:
29:                   G_edge.encode(edge);
30:                 else id ← E.retrieveID(edge);
31:                     E.update(edge);
32:                 end if
33:         end if
34          previous ← current
35:         current ← destination;
36: end while
```

5. Example

An example of how algorithm 1 models and encodes T = "Democracy means a Government of the people, by the people, for the people" is shown in figure 1. The function performed in each step is shown under the graph model which represents the current word net state. A single word is processed in each step, so a specific edge is followed for encoding purposes. Each vertex contains the word that it stores and its global identifier in the word net. In turn, each edge shows two values (n:b): n represents the distance between the two words (we are considering the maximum value of n *i.e.* n_{max}=2) and b is the is the number of times that the edge has been traversed. Additionally, the state of the streams G_{text}, G_{vertex}, and G_{edge} is progressively updated. The process begins when the first word, "Democracy", is processed. The word net is currently empty, so a new vertex (identified as 1) is appended to store "Democracy". The word "Democracy" is appended in G_{text}. No edges are appended because this is the first word in T. This new vertex plays the current role for the next step. The single space is omitted because this modeling is performed in conjunction with the spaceless words transformation. The word "means" is next processed (step 2). This is its first occurrence, so a new vertex (identified as 2) and a new edge (identified as 1:1) are appended. This edge represents the word transition currently traversed (from "Democracy" to "means"). It is identified in the source vertex as 2, and is initialized with single occurrence (this information is represented by the pair 1:1 over the edge). The word "means" is appended in G_{text}. 2 is the current vertex for the next step. Similar functions are performed for the following words "a" ,"Government", "of " , 'the', 'people' , ',' and 'by' (Steps 3 to 9).

Step 1:

G_{text}	Democracy
G_{vertex}	1
G_{edge}	N/A

Graphical representation(in sparse matrix form)
N/A

Step 2:

G_{text}	Democracy	means
G_{vertex}	1	2
G_{edge}	N/A	1.1

Graphical representation(in sparse matrix form)

0 1.1

0 0

Step 3:

Step 4:

G_{text}	Democracy	means	a
G_{vertex}	1	2	3
G_{edge}	N/A	1.1	1.1, 2.1

Graphical representation(in sparse matrix form)

0 1.1 2.1
0 0 1.1
0 0 0

G_{text}	Democracy	means	a	govornment
G_{vertex}	1	2	3	4
G_{edge}	N/A	1.1	1.1, 2.1	1.1, 2.1

Graphical representation(in sparse matrix form)

0 1.1 2.1 0
0 0 1.1 2.1
0 0 0 1.1
0 0 0 0

Step 5:

G_{text}	Democracy	means	a	govornment	of
G_{vertex}	1	2	3	4	5
G_{edge}	N/A	1.1	1.1, 2.1	1.1,2.1	1.1, 2.1

Graphical representation(in sparse matrix form)

0 1.1 2.1 0 0
0 0 1.1 2.1 0
0 0 0 1.1 2.1
0 0 0 0 1.1
0 0 0 0 0

Step 6:

G_{text}	Democracy	means	a	govornment	of	the
G_{vertex}	1	2	3	4	5	6
G_{edge}	N/A	1.1	1.1, 2.1	1.1,2.1	1.1,2.1	1.1, 2.1

Graphical representation(in sparse matrix form)

0 1.1 2.1 0 0 0
0 0 1.1 2.1 0 0
0 0 0 1.1 2.1 0
0 0 0 0 1.1 1.2
0 0 0 0 0 1.1
0 0 0 0 0 0

Step 7:

G_{text}	Democracy	means	a	govornment	of	the	people
G_{vertex}	1	2	3	4	5	6	7
G_{edge}	N/A	1.1	1.1, 2.1	1.1,2.1	1.1, 2.1	1.1,2.1	1.1, 2.1

Sparse matrix omitted

Step 8:

G_{text}	Democracy	means	a	govornment	of	the	people	,
G_{vertex}	1	2	3	4	5	6	7	8
G_{edge}	N/A	1.1	1.1, 2.1	1.1,2.1	1.1,2.1	1.1,2.1	1.1,2.1	1.1, 2.1

Sparse matrix omitted

Step 9:

G_{text}	Democracy	means	a	govornment	of	the	people	,	by
G_{vertex}	1	2	3	4	5	6	7	8	9
G_{edge}	N/A	1.1	1.1, 2.1	1.1,2.1	1.1,2.1	1.1,2.1	1.1,2.1	1.1, 2.1	1.1, 2.1

Sparse matrix omitted

At this moment, 9 (Step 10) is the current vertex and the word "the" is now processed. This is not its first occurrence. It is represented in the vertex 6. However, the edge (9, 6) not belongs to E, so a New Edge must be appended. As can be seen in Step 10, this new edge (9, 6) is appended to represent the new word transition from "by" to "the". 6 evolve to the current vertex.

The word "people" is next processed (Step 11).The edge (6,7) $\in E$. Therefore only the statistical data will be updated. 7 evolves to the current vertex. The punctuation mark "," is next processed (step12). The edge (7,8) $\in E$ and only the statistical data will be updated. 8 evolves to the current vertex.

The process continues with the processing of a new word, "for", (step 13) and a new transition between "," and "for(10)" . At this moment '10' is the current vertex and the word the is now processed. This is not its first occurrence. It is represented in the vertex 6. However, the edge (10, 6) not belongs to E, so a New Edge must be appended. 6 evolves to the current vertex. The word "people" is next processed(Step 15). The edge (6,7) $\in E$. Therefore only the statistical data will be updated. 7 evolves to the current vertex. The punctuation mark "." is next processed (step16). Finally The process ends with the processing of a new word, ".", (step 16) and a new transition between "people" and "(11)" . It is to be noted that the above discussion is limited to only first order (n=1) edges.

Step 10:

G_{text}	Democracy	means	a	govornment	of	the	people	,	by
G_{vertex}	1	2	3	4	5	6	7	8	9
G_{edge}	N/A	1.1	1.1, 2.1	1.1,2.1	1.1,2.1	1.1,2.1 1.1,2.1	1.1,2.1	1.1,2.1	1.1, 2.1

Graphical representation(in sparse matrix form)

0	1.1	2.1	0	0	0	0	0	0
0	0	1.1	2.1	0	0	0	0	0
0	0	0	1.1	2.1	0	0	0	0
0	0	0	0	1.1	2.1	0	0	0
0	0	0	0	0	1.1	2.1	0	0
0	0	0	0	0	0	1.1	2.1	0
0	0	0	0	0	0	0	1.1	2.1
0	0	0	0	0	2.1	0	0	1.1
0	0	0	0	0	1.1	0	0	0

Step 11:

G_{text}	Democracy	means	a	govornment	of	the	people	,	by
G_{vertex}	1	2	3	4	5	6	7	8	9
G_{edge}	N/A	1.1	1.1, 2.1	1.1,2.1	1.1,2.1	1.1,2.1 1.1,2.1	1.2,2.1 2.1	1.1,2.1	1.1, 2.1

Sparse matrix omitted

Step 12:

G_{text}	Democracy	means	a	govornment	of	the	people	,	by
G_{vertex}	1	2	3	4	5	6	7	8	9
G_{edge}	N/A	1.1	1.1, 2.1	1.1,2.1	1.1,2.1	1.1,2.1 1.1,2.1	1.2,2.1 2.1	1.2,2.2	1.1, 2.1

Sparse matrix omitted

Step 13:

G_{text}	Democracy	means	a	govornment	of	the	people	,	by	for
G_{vertex}	1	2	3	4	5	6	7	8	9	10
G_{edge}	N/A	1.1	1.1, 2.1	1.1,2.1	1.1,2.1	1.1,2.1 1.1,2.1	1.2,2.1 2.1	1.2,2.2	1.1, 2.1	1.1, 2.1

Sparse matrix omitted

Step 14:

G_{text}	Democracy	means	a	govornment	of	the	people	,	by	for
G_{vertex}	1	2	3	4	5	6	7	8	9	10
G_{edge}	N/A	1.1	1.1, 2.1	1.1,2.1	1.1,2.1	1.1,2.1 1.1,2.2, 1.1	1.2,2.1 2.1	1.2,2.2	1.1, 2.1	1.1, 2.1

Sparse matrix omitted

Step 15:

G_{text}	Democracy	means	a	govornment	of	the	people	,	by	for
G_{vertex}	1	2	3	4	5	6	7	8	9	10
G_{edge}	N/A	1.1	1.1, 2.1	1.1,2.1	1.1,2.1	1.1,2.1 1.1,2.2, 1.1	1.3,2.1 2.1,1.1	1.2,2.2	1.1, 2.1	1.1, 2.1

Sparse matrix omitted

Step 16:

G_{text}	Democracy	means	a	govornment	of	the	people	,	by	for	.
G_{vertex}	1	2	3	4	5	6	7	8	9	10	11
G_{edge}	N/A	1.1	1.1, 2.1	1.1,2.1	1.1,2.1	1.1,2.1 1.1,2.2, 1.1	1.3,2.1 2.1,1.1	1.2,2.2	1.1, 2.1	1.1, 2.1	1.1, 2.1

Graphical representation(in sparse matrix form)

0	1.1	2.1	0	0	0	0	0	0	0	0
0	0	1.1	2.1	0	0	0	0	0	0	0
0	0	0	1.1	2.1	0	0	0	0	0	0
0	0	0	0	1.1	2.1	0	0	0	0	0
0	0	0	0	0	1.1	2.1	0	0	0	0
0	0	0	0	0	0	1.3	2.2	0	0	1.1
0	0	0	0	0	0	0	1.2	2.1	2.1	2.1
0	0	0	0	0	2.2	0	0	1.1	1.1	0
0	0	0	0	0	1.1	1.1	0	0	0	0
0	0	0	0	0	1.1	2.1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

Figure 1. $CSGM_2$ in Compression: Word Net Building and Encoding. (Step1-16)

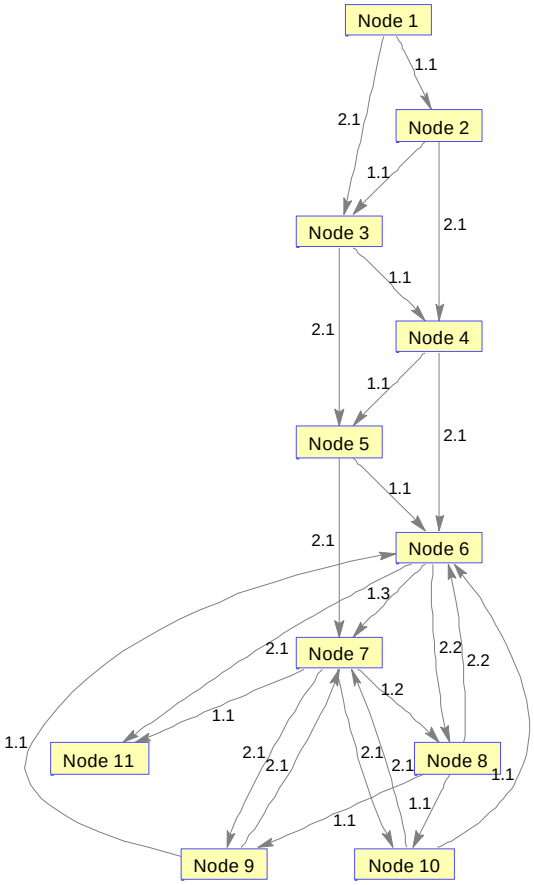


Figure 2. Graph Generated using MATLAB for the Sentence "Democracy Means a Government of the People, by the People, for the People." from the Sparse Matrix Obtained, in Accordance with $CSGM_2$ Model

Hence the $CSGM_2$ has reduced the sentence "Democracy means a government of the people, by the people, for the people." to "Democracy means a government of the people, by for.". The technique has reduced 19 characters in one sentence only, at the cost of the sparse matrix, which is approximately 19*8=152 bits (ignoring the space occupied by the

sparse matrix). According to Heaps law the vocabulary grows sublinearly with the collection size. It can be formulated as $V_R(n) = Kn^{\beta}$ where V_R is the number of distinct words in an instance text of size *n*. *K* and β are free parameters determined empirically. With English text corpora, typically the minimum value of *K* is 10, and minimum of β is 0.4. Considering the minimum values, if we think about a text file having 300 words then the number of distinct words will be approximately 98 which is about 300% less than the original file. This precision can only be achieved with the help of graph representation at the cost of the space required to store the sparse matrix. This also implies that for small size texts the memory occupied by the graph representation may larger or equal to the text representation. But as the text file size increases the effectiveness of $CSGM_2$ also increases.

6. Conclusion

This paper proposes and confirms a preprocessing step for data compression that can be used to optimize all the existing data compression algorithms. The above graph based approach describes a specific preprocessing technique for natural language text compression. The text transformation (word net), which $CSGM_2$ approaches, obtains an impressive representation that can be built in an effective way through a word-based directed graph. The resultant model gives a 2^{nd} order text representation based on the information provided by the edges in the word net. These edges are dynamically handled on some heuristics. Thus, each word transition in the original text is encoded with the help of the local identifier of each edge in its respective source vertex.

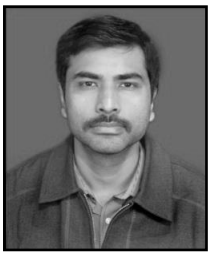
In this paper it was desired to provide a new strategy for data compression. As the world is moving towards the goal of providing highest service at a lowest expense, this strategy will make any text segment able to use lesser memory space without decreasing its features rather will increase its usability and portability. It will decrease the memory area occupied by text segment in any type of file and also speed up its transfer time through the internet.

References

[1] H. Altarawneh and M. Altarawneh, "Data Compression Techniques on Text Files: A Comparison Study", International Journal of Computer Applications, vol. 26, no. 5, (**2011),** July.
[2] R. Baeza-Yates and B. Ribeiro-Neto, "Modern Information Retrieval", ACM Press / Addison-Wesley, (**1999**).
[3] R. Horspool and G. Cormack, "Constructing word-based text compression algorithms", in: Proc. of IEEE Data Compression Conference, (**1992**).
[4] D. Salomon, "Data Compression", Springer, (**2004).**
[5] K. K. Phukon, "A Composite Graph Model for Web Document and the MCS Technique", International Journal of Multimedia and Ubiquitous Engineering, vol. 7, no. 1, SERSC, **(2012)** January.
[6] H. Heaps, "Information Retrieval - Computational and Theoretical Aspects", Academic Press, (**1978**).
[7] D. Huffman, "A method for the construction of minimum-redundancy codes",in: Proc. of the Institute of Electronics and Radio Engineers, vol. 40, (**1952**).
[8] A. K. Azad, R. Sharmeen, S. Ahmad and S. M. Kamruzzaman, "An Efficient Technique for Text Compression", in: Proc. of The 1st International Conference on Information Management and Business (IMB2005).
[9] J. Ziv and A. Lempel, "An universal algorithm for sequential data compression", IEEE Transactions on Information Theory, vol. 23, (**1977**).
[10] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding", IEEE Transactions on Information Theory, vol. 24, (**1978**).
[11] B. Langmead, "Introduction to the Burrows-Wheeler Transform and FM Index", Department of Computer Science, JHU, (**2013**) November 24.
[12] M. A. Martinez-Prieto, J. Adiego and P. de la Fuente, "Natural Language Compression on Edge-Guided Text Preprocessing", in: Proc. of 14th International Symposium on String Processing and Information Retrieval (SPIRE'07), (**2007**), pp. 14–25.
[13] A. Moffat, "Word-based text compression", Software Practice & Experience, vol. 19, (**1989**), pp. 185–198.

[14] J. Culpepper, "Efficient Data Representations for Information Retrieval", Ph.D. thesis, Department of Computer Science and Software Engineering, University of Melbourne, Australia, (**2008**).
[15] V. Nastase and S. Szpakowicz, "Matching Syntactic-Semantic Graphs for Semantic Relation Assignment", in: Proc. of TextGraphs: Graph-based Algorithms for Natural Language Processing.
[16] N. Deo, "GRAPH THEORY with Applications to Engineering and Computer Science", PHI Learning Private Limited, ISBN-978-81-203-0145-0.

Authors



Kaushik Kishore Phukon, Assistant Professor (IT), Department of Commerce, Gauhati University, India received MCA degree from Jorhat Engineering College (Under Dibrugarh University), India in 2009. He is currently pursuing doctoral research at Department of Computer Science, Gauhati University, Assam, India. His research interests include representation of web documents using graphs and graph based clustering and classification.

Hemanta K. Baruah, V.C., Bodoland University and former Dean Faculty of Science, Gauhati University is a Reviewer in the Mathematical Reviews, USA. He received his M.Sc. Degree in Statistics from Gauhati University, Guwahati, India, in 1975, and his Ph.D. degree in Mathematics from Indian Institute of Technology, Kharagpur, India, in 1980. He is a Professor in the Department of Statistics, Gauhati University, since 1995. His research interests are in mathematics of fuzziness, graph theory, data mining and mathematical modeling.

