AUTOMATIC SHORT MAIL CONSTRUCTION TO ENHANCE EMAIL STORAGE

Nijin.V[1], Kaushik.Ra[1], Swathika.R[2], Mohanavalli.S[2]

[1]UG Student, Dept. of Information Technology, SSN College of Engineering
nijinvinodan4@gmail.com, kaushikramkumar4@gmail.com
[2]Assistant Professor, Dept. of Information Technology,
SSN College of Engineering
swathikar@ssn.edu.in, mohanas@ssn.edu.in

ABSTRACT

Short Texts play a significant role in reviews, blogs, social networking sites etc. The replacement of featured text by short text can greatly reduce the text size, thereby allowing more information to be communicated. Emails apparently do not have large number of short texts. This paper proposes a novel idea for automatic construction of Short Text Dictionary (STD) that maps the large texts into their short texts thereby curtailing the storage space efficiently. An algorithm has been proposed to convert the featured word into a new Short Word. Normalisation in Sentimental Analysis involves pre-processing of short texts. This constructed dictionary with transliterated word processing can thus be used as a pre-processing tool.

KEYWORDS

Short Mail, Short Word, Short Text Dictionary (STD), WordNet.

1. INTRODUCTION

It has been estimated that 294 billion mails are sent everyday with 2.8 million mails sent every second and about 90 trillion per year which also includes spam. These escalating mails sent create an adverse impact on storage. In this paper, we have proposed a novel idea which would increase the storage efficiency. E-mails can be broadly classified into two categories namely- formal mails & informal mails. Formal mails have long text words i.e. the featured words (For example: *Meet you tomorrow*). The latter category includes short text words (For example: *meet u tmrw*). Depending on the size of the mail, the storage capacity differs.

Assuming that *65 percentage* of the mails contain words that are neither clipped nor short texted, we have efficiently tried to store those mails as *Short Mails* (We introduce a term to describe converted short text mail) and then later reverse map it to produce the original text when being viewed by the recipient. Therefore it requires pre-processing to be performed on the mail before storage. This seems to be time consuming but the overall storage efficiency gained would outweigh the delay. The concept of encryption [6] is not taken into account as we process it on

David C. Wyld (Eds) : ICCSEA, SPPR, CSIA, WimoA - 2013
pp. 49–56, 2013. © CS & IT-CSCP 2013 DOI : 10.5121/csit.2013.3505

raw data. Further encryption on the data also creates a new word which would have its own related short clipped word in the dictionary. The text compression algorithms [5] [4] can be applied on these short texts which will once again compress the text. Majority of the mails sent, follow common dictionary words with varying size. In this paper, a message or mail is analysed as series of words rather than processing it entirely. An initial dictionary of Short Words can be constructed over the WordNet [3]. Each word in the dictionary has a unique *Short Word* (We introduce another term to describe clipped words). The algorithm would automatically form short words for the terms, if any new inclusions happen in the Wordnet. This is similar to the one as proposed by Bo Han et.al [1] [12] but in reverse manner. Bo Han mapped *tmrw* as tomorrow in the analysis of microblogs.

The rest of this paper is organized as follows: Section 2 describes the proposed work; section 3 explains the practical experiments conducted; section 4 highlights the performance and section 5 enumerates the conclusion and the future work.

2. PROPOSED WORK

The Short Words are created by extracting patterns from the given word. We have maintained the featured text for words of length 4 or smaller and all other words violating this *threshold* are converted into Short Words. The constraint on the length is maintained to preserve the identity of the word. The uniqueness of each newly created Short Word is necessary as it is highly important while converting it back to their featured text. *Fig.1* depicts the proposed work.

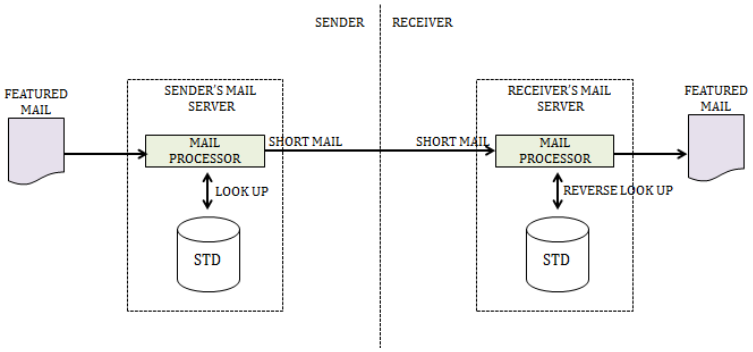

Fig. 1. Working Procedure of Short Mails

The following steps occur when a user creates and sends an email.

Step1: Compose a mail

Step2: The sender sends the mail (Assuming all other security measures are performed by cryptographic algorithms).

Step3: When the mail reaches the mail box of the mail service provider, it is converted into Short Mail by comparing the words with STD and then stored.

On the receiver side, the mail is stored as Short Mail and when the recipient opens the mail, it is converted into actual content by looking up into the STD. Thus the actual content is preserved and communicated without loss.

2.1 STD Construction

In this section we focus on the task of constructing a dictionary containing a list of featured and Short Words. STD can be initially constructed by taking words from the WordNet or it can be automatically constructed. As it encounters new word, the dictionary is made to learn to generate new Short Word for the corresponding actual word. We define our Dictionary D as a collection of paired words (Featured Word or Actual word (AW), Short Word (SW)) i.e. *D = (AW, SW)* similar to the one proposed in [2]. STD would then build up over a period of time and it will then contain Short Words for majority of the featured words. STD would serve the purpose of a reusable repository which could be applied in different domains of Text Mining. Dictionary can be constructed using Hash Table or B-Trees [7]. These data structures are much preferred when compared to other existing data storage concepts because B-trees and Hash Table generally have an average complexity of *O (log n)* and *O (1)* for insertion and searching respectively.

2.2 Construction of Short Mail

Each word in the mail is analysed separately. The mail is divided into series of subsets (each subset will contain a word). The words which are greater than the *threshold* are separated from other mail contents for generating Short Word. The words which are lesser than the *threshold* are not considered as most of them are prepositions, adjectives, auxiliary verbs. Moreover the Short Words generated for such words will be extremely small which would create problems during remapping. *Fig. 2* explains the creation of Short Mail.

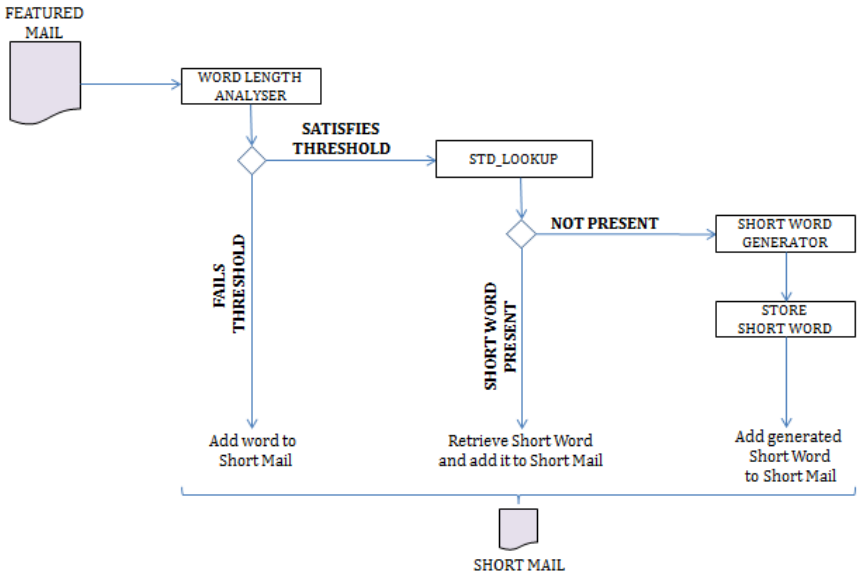

Fig. 2. Flowchart explaining construction of Short Word and Short Mail

Initially a word in the mail is checked for its length. If it is less than or equal to the *threshold*, then it is automatically appended to the Short Mail. Otherwise the words that are greater in length are processed separately. The featured word is first checked with the Dictionary and if a short text pertaining to the word is present, then that Short Word is concatenated to the Short Mail.

In case of new words, the algorithm would generate a Short Word for it, which is stored in the dictionary and also added to the Short Mail. The same procedure is repeated for all the words in the mail. The Short Mail is then stored and sent.

2.3 Generation of Short Word

We propose a simple algorithm which translates the featured word to a Short Word. The Short Words are about half the length of the featured word. The Short words are formed by taking the letters at the odd position of the word. For example: *tomorrow – tmro*. For words with even number of letters, length of the short Word is *n/2* and for odd number of letters, length of the Short Word is *(n+1)/2*. If the generated short words are similar, then the algorithm generates extra bit and concatenates it to the new Short Word to differentiate it. For example: The words *india*, *indian*, *indias* all have *ida* as Short Word according to the previous algorithm. This may degrade the efficiency while remapping. To avoid this, the algorithm generates *ida* for india, *ida0* for indian and *ida1* for indias and so on.

For every Featured Word or Actual Word (AW), the Short Word (SW) is generated by,

$$length(sw) = \frac{length(AW)}{2}, \quad \text{if } length(AW) \text{ is even} \tag{1}$$

$$length(sw) = \frac{length(AW)+1}{2}, \quad \text{if } length(AW) \text{ is odd} \tag{2}$$

$$length(sw) = length(AW), \quad \text{if } length(AW) <= threshold \tag{3}$$

3. EXPERIMENTS

We have created our own mail service (*nibble*) which reduces the content of the mail before storing it. Assuming that the STD has few featured and their corresponding Short Words at present (*Table 1*). The Dictionary has two columns to store the Featured text and Short Word. This Dictionary will learn automatically over a period of time. Insertion and search can be made quicker by indexing the columns in the Dictionary.

Table 1. Initial List of Featured and Short Words in STD

INDEX	FEATURED WORD	SHORT WORD
1	tomorrow	tmro
2	process	poes
3	india	ida
4	indian	ida0
5	wonderful	wnefl
6	experience	eprec

The mailing procedure and its corresponding short mail conversion and generation are explained below.

Step 1: The user composes the mail. The message is "*Tomorrow is going to be a splendid day*"
Step 2: When the user clicks send, the dictionary is referred to find out the short text for words greater than the threshold. If the dictionary has a short text for a given featured word, then its

short text is added to the Short Mail and the other Short Words are generated by the conversion algorithm and then stored in the dictionary and appended to the Short Mail.

Here Short Mail is represented as *shortMail* for our implementation purpose.
Initially, *shortMail* = " "

i. The word "*tomorrow*" already has a Short Word "*tmro*" in the Dictionary and so it is added to Short Mail string.
Now, *shortMail* = *tmro*

ii. "*is*" is a smaller word and so it is directly concatenated to the Short Mail.
Therefore *shortMail* = *tmro* +*is* +

iii. "*going*" is not present in the dictionary and so a Short Word "gig" is generated for it. "*gig*" is stored in the Dictionary and then added to short Mail and so on.

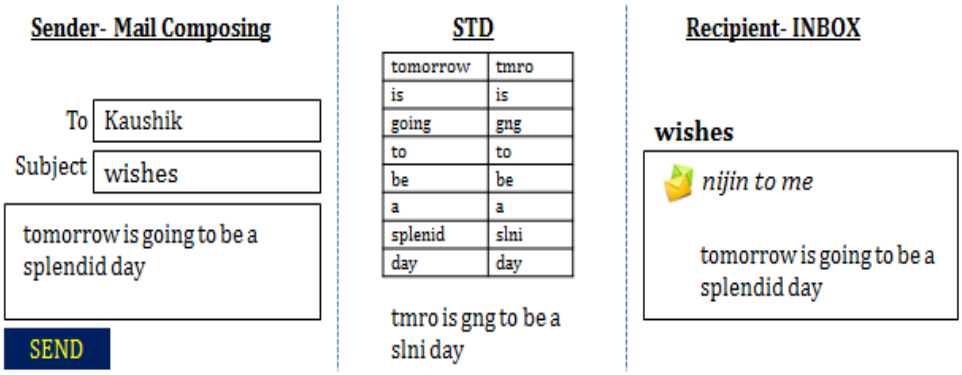

Fig. 3. Composing and receiving of mail in nibble

The spaces and punctuations are automatically taken care by the conversion algorithm.
Finally, *shortMail* = *tmro* +*is* +*gig* +*to* +*be* +*a* +*slni* +*day*

Table 2. Snapshot of STD after new words are added

INDEX	FEATURED WORD	SHORT WORD
1	tomorrow	tmro
2	process	ppoes
3	india	ida
4	indian	ida0
5	wonderful	wnefl
6	experience	eprec
7	*going*	*gng*
8	*splendid*	*slni*

This Short Mail is now stored into the Email database as "*tmrw is gng to be a slni day*". When the receiver pulls his mail into the inbox, the reverse algorithm transfers the Short Mails into the actual content again by looking up in the STD. When the recipient opens the mail, the actual content is displayed to him (*Fig. 3*).

4. PERFORMANCE

The results and performances were drawn from three different cases. The first case (*Table 3*) involved testing STD to handle the same set of words repeatedly, the second case (*Table 4*) dealt with different optimized set of words and the final case (*Table 5*) had both repeated and unrepeated words. Graphs had been plotted against execution time to show that the delay caused is extremely small. Performance on an average saved 50% of the storage by converting the mails into Short Mails.

Case 1: When tested on message content containing 112 words of length greater than the *threshold* with 999 letters, the mail content was reduced to 538 letters which gained an efficiency of 53.853%. As the words were repeated, the delay caused was reduced further and became constant over a period of time.

Table 3. Processing Time when input set has repeated words

Trial ID	Dictionary Status	No.of Words	No.of Letters	Short Mail Length	Processing Time (milliseconds)
E	Empty	111	999	538	3541
T1	111 words	111	999	538	1029
T2	same 111 words	111	999	538	764
T3	same 111 words	111	999	538	749
T4	same 111 words	111	999	538	733
T5	same 111 words	111	999	538	749

Case 2: When different set of inputs without repetition of words was tested, the time taken to generate Short Mail remained constant and relatively high when compared to previous case. This case proved that as the words are new to STD, the time taken to convert and store words is high.

Table 4. Processing Time when input set has new words

Trial ID	Dictionary Status	No.of Words	No.of Letters	Short Mail Length	Processing Time (milliseconds)
E	Empty	107	997	532	2801
T1	107 words	107	998	528	2016
T2	214 words	109	999	535	2004
T3	323 words	92	999	544	2139
T4	415 words	127	999	544	2384

Case 3: In this, we have tested the Dictionary with the combination of repeated and unrepeated words. Here the time taken to process was found to be an intermediate value compared to the previous cases. This test drew the final conclusion explaining that the delay on conversion is entirely based on the occurrence of new words. Also when T1 was repeated again as T5, the processing time taken was found to vary only by few fractions.

Table 5. Processing Time when input set has both repeated & new words

Trial Id	Dictionary Status	No.of Repeated words (letters)	No.of New words(letters)	Short Mail Length	Processing Time (ms)
T1	415 words	65(524)	61(543)	543	1453
T2	476 words	61(461)	62(538)	538	1384
T3	538 words	62(597)	54(498)	535	1402
T4	592 words	57(463)	59(533)	542	1397
T5	651 words	65(524)	61(475)	543	1451

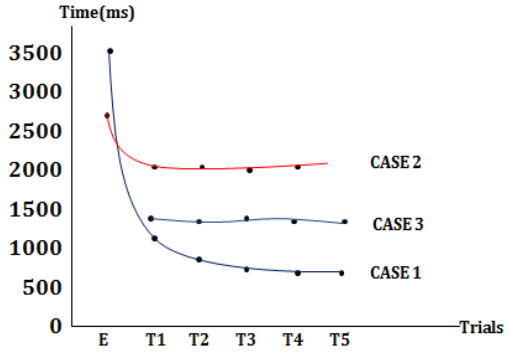

Fig. 4. Processing Time Comparison for each case

Fig. 4 shows a graph that compares all the three cases explaining the time delay occurred when different combination of words are used. It shows that when the words are repeated the delay is less (*Case 1*) and when words are new each time, the delay caused is very high (*Case 2*) and finally when combinations of new and old words are used, the delay is intermediate (*Case 3*). Also, as STD learns automatically, it is assumed that at one instant of time when STD had learned enough, the delay caused in conversion will be negligible. However, the capacity of the Mail storage remained to be half of the original capacity in each case thereby supporting the concept of the proposed algorithm.

5. CONCLUSION AND FUTURE WORKS

In this paper, we have proposed an algorithm for the efficient storage of electronic mail. From the experiments conducted using raw data on the Dictionary (STD), it has been proved that an efficient storage could be achieved if the mails are stored as Short Mails. Furthermore the dictionary has the capability to analyse the content and generate new short words. This novel idea paves the way for the cryptographic and compression algorithm technique to perform their normal security measures efficiently.

Our future work involves the reverse conversion of short texts into actual vocabulary words with transliterated word processing [11] which will be useful for Sentimental analysis to replace the traditional Normalisation algorithm.

REFERENCES

[1] Bo Han, Paul Cook and Timothy Baldwin. 2012. Automatically Constructing a Normalisation Dictionary for Microblogs. In Proceedings of the Conference on Empirical Methods in Natural Language Processing and Natural Language Learning (EMNLP-CoNLL 2012), pages 421–432, Jeju, Korea.

[2] Shantanu Godbole, Indrajit Bhattacharya, Ajay Gupta, Ashish Verma, "Building re-usable dictionary repositories for real-world text mining". CIKM 2010: 1189-1198.

[3] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross and Katherine J. Miller, "Introduction to WordNet- An On-line Lexical Database",1993.

[4] Lempel - ZIV welch coding - Terry Welch, "A Technique for High-Performance Data Compression", IEEE Computer, June 1984, p. 8–19.

[5] Huffman coding - D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., September 1952, pp 1098–1102. Huffman's original article.

[6] Encryption-Helen Fouché Gaines,"Cryptanalysis", 1939, Dover. ISBN 0-486-20097-3.

[7] M.A. Weiss- Data Structures and Algorithms, ISBN: 0-321-37013-9.

[8] Songbo Tan, Qiong Wu Key, "A random walk algorithm for automatic construction of domain-oriented sentiment lexicon", Expert Systems with Application, Volume 38 Issue 10, September 2011, Pages 12094-12100.

[9] Inga Gheorghita, Jean-Marie Pierrel, "Towards a methodology for automatic identification of hypernyms in the definitions of large-scale dictionary", In Proceedings of Gheorghita12.531

[10] Nobuhiro Kaji and Masaru Kitsuregawa, "Building Lexicon for Sentiment Analysis from Massive Collection of HTML Documents", In Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning EMNLP-CoNLL (2007) , p. 1075—1083

[11] Kevin Knight and Jonathan Graehl, "Machine Transliteration", Information sciences Institute. University of California. Maria del rey, CA 90292.

[12] Bo Han and Timothy Baldwin, "Lexical Normalisation of Short Text Messages: Makn Sens a #twitter", Association for Computational Linguistics, p 368-378.