Development of Word-Based Text Compression Algorithm for Indonesian Language Document

Ardiles Sinaga, Adiwijaya
Telkom University
Bandung, Indonesia
sinaga.diles@gmail.com,
adiwijaya@telkomuniversity.ac.id

Hertog Nugroho
Telecommunication Engineering
Bandung State of Polytechnic
Bandung, Indonesia
hertog@polban.ac.id

***Abstract*— Information technology is growing very rapidly, in particular for data handling. Data is a valuable asset for everyone, especially for larger companies with branches in several places. Data transmission from headquarters to branch offices make the company must provide good tools to do it. These companies also need tools that can be used to compress data to reduce their size.**

The main idea of the word-based encoding is to extract each word of the source text, then it is checked whether containing capital letters or not. After that, it is checked if there is a symbol or number. The particle will be separated from the basic word using stemming algorithm. Symbols, numbers and affixes will be indexed in the basic dictionary. The basic word will also be checked whether it exists in the basic dictionary or not. If there is not a match, then the word will be stored in the supplement dictionary.

The experiment was conducted on the text file with the size from about 10K bytes up to 500K bytes with 16-bits length codewords. The result shows that the compression ratio of the proposed method is comparable with the previous ones, while its processing time is much better than the Reversed Sequence of Characters on LZW method.

***Keywords*— Data Compression, Word-Based, LZW, Stemming, Tree Structure**

I. INTRODUCTION

A study of data compression methods with possible application to optimization of data transmission process is very important. In computer science and information theory, data compression involves encoding information using fewer bits than the original representation. Data compression is useful because it helps reduce resources usage, such as data storage space or transmission capacity. Many researchers have developed compression methods to increase the compression ratio. One of them was developed by Lempel, Ziv and Welch [4]. This compression technique uses dictionary to store the codeword(s) which is/are used for encoding and decoding data. Another study using the basic idea of LZW technique is the WB-LZW which was developed by Horspool and Cormack [1]. They developed a word-based compression technique which was very different from the standard character-based LZW compression technique.

TABLE I. VARIOUS TEXT COMPRESSION PERFORMANCE

Document Name	Original Size in bytes	Relative Size After Compression				
		Standard LZW	*WB-Adaptive Huffman*	*WB-LZW*	*State-Based*	*WB- First Order*
On-line manual page for *csh*	66772	40.40%	29.80%	34.30%	21.60%	29.00%
On-line manual page for *make*	63761	42.70%	34.70%	35.80%	29.50%	31.70%
LaTex file	83106	44.70%	36.10%	32.00%	36.00%	33.70%
C Source File	24706	39.20%	28.50%	26.40%	32.30%	25.30%

Table I shows performance comparison of several text compression methods. WB-Adaptive Huffman algorithm [2] used the technique of calculating the estimated probability of words that appear most frequently and store it into the dictionary which is designed based on the number of words that appear most frequently. The most frequently appearing words will be encoded and stored in the dictionary that has the smallest number of bits. However, this algorithm has disadvantage on the compression performance. It has slow speed compression performance.

WB-LZW algorithm [1] used two dictionaries to store the codeword of each string. The first dictionary was used to store the code of the word and the second was used to store the code of the symbol. Performance of compression was very suitable for small files, but it wasn't for large files. It is caused by a number of words and symbols that exist in the file, so it will affect the size of the dictionary.

WB-State-Based [1] used modeling to generate the source text and use the prediction probabilities by an arithmetic coding subroutine. It is used to predict the relationship between Article, Noun, Adjective, Verb and others.

WB-First Order [1] used an algorithm to predict the words that will appear and the words that appear most frequently. The assumption was that there should be a strong correlation between the words and the alphanumeric words in the file. The

technique also used statistic to reduce the volume of data that must be retained for similar words.

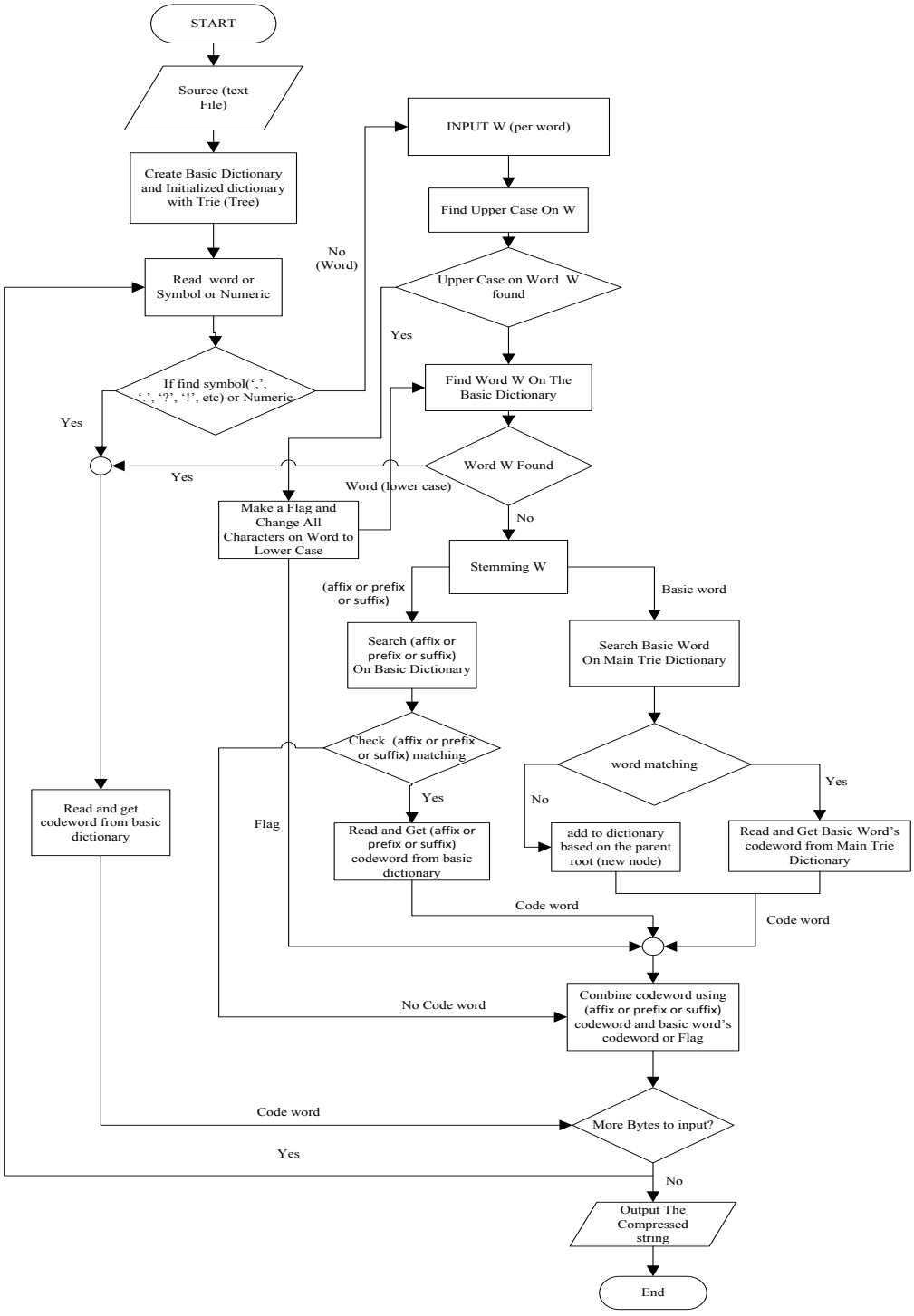

Fig. 1. The proposed Modified Word Based Text Compression Algorithm for Indonesian Language Document

II. MODIFIED WORD BASED TEXT CCOMPRESSION ALGORITHM

In the proposed method, we have used the basic dictionary to store the words that most frequently appear, such as conjunctions, pronouns, affixes, prefixes, suffixes, symbols and numbers. Moreover this proposed method also uses the separation of uppercase and lowercase letters.

This study proposed a new approach to modify the word-based text compression algorithm for Indonesian language document from the basic WB-LZW (Word Based Lempel Ziv Welch) Compression algorithm [1]. The main objective is to increase the compression ratio by maximizing the use of dictionary to store the words and the codewords and to speed up the word search time from dictionary.

Figure 1 shows the general flow of the algorithm. Read the source text, and then extract each word of the source text, then it is checked whether containing capital letters or not. After that, it is checked if there is a symbol or number. The particle will be separated from the basic word using stemming algorithm. Symbols, numbers and affixes will be indexed in the basic dictionary. The basic word will also be checked whether it exists in the basic dictionary or not. If there is not a match, then the word will be stored in the supplement dictionary.

The proposed method creates a supplement dictionary in a trie [12] that is called prefix tree to speed up the word searching in the dictionary.

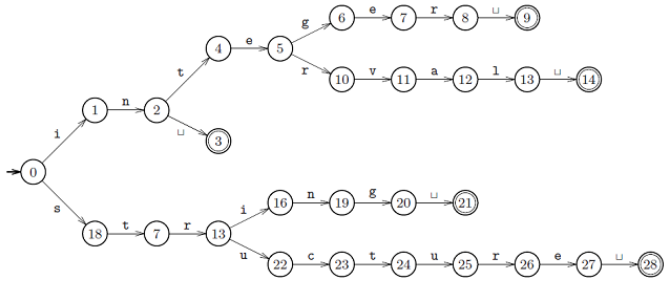

Fig. 2. Trie of X = {*in, integer, interval, string, structure*}.

The algorithm IsInTrie(*root*, *x*) (see Figure 2), tests if the string x is present in the trie and consequently if the string *x* is a prefix of strings represented by the trie. It works, similarly to the creation of the trie, by spelling, from the *root* of the trie, the longest prefix of *x* corresponding to a branch in the trie. If this longest prefix is *x* itself, then the algorithm returns true and the string *x* belongs to the trie, otherwise the algorithm returns false and the string is not a prefix of any string in the set. The algorithm IsInTrie(*root*, *x*) works in time $O(|x|)$ or $O(|x| \times \log\ \sigma)$ depending on the branching time.

In addition, this study proposed a method that allows the use of stemming methods[5][6] to separate the word from its affix. This is to maximize the use of the dictionary, since it is not necessary to keep all the words containing the same basic words. It only keeps the stemming words in the dictionary. The proposed method used a basic dictionary to store affixes (prefixes and suffixes), pronouns, conjunctions, and symbols and supplement dictionary to store basic word (without affix, prefix and suffix).

A. Creating New Codeword

Some requirements are needed to create a new codeword which will be stored into the supplement dictionary. The detail requirements to create the new codewords are as follows:

1) Stemming Rules

- Rules and Variations Form of Prefixes

TABLE II. EXAMPLE VARIATIONS FORM OF PREFIXES [6]

Prefix	Variation Form	Rules
meng	meng	+ Vowel|k|g|h…, e.g: *ambil* (to take) → *mengambil* (taking) *embun* (vapor) → *mengembun* (to condense) *ikat* (to tie) → *mengikat* (to tie/to bind) *olah* (process) → *mengolah* (processing) *ukur* (to measure) → *mengukur* (measuring) *kurus* (slim) → *mengurus* (become slimmer) *urus* (to take care) → *mengurus* (to take care) *ganggu* (to disturb) → *mengganggu* (disturbing) *hilang* (to lose) → *menghilang* (to dissapear)
	meny	+ s…, e.g: *sisir* (comb) → *menyisir* (to comb something)
	mem	+ b|f|p…, e.g: *beku* (frozen) → *membeku* (to become frozen) *fitnah* (to accuse) → *memfitnah* (accusing) *pukul*(to hit) → *memukul* (hitting)
	men	+ c|d|j|t… *cuci* (to wash) → *mencuci* (washing) *darat* (land) → *mendarat* (landing/docking) *jual* (to sell) → *menjual* (selling) *tukar* (to change) → *menukar* (changing)
	me	+ l|m|n|r|y|w…, e.g: *lintas* (to cross) → *melintas* (crossing) *makan* (to eat) → *memakan* (eating) *nikah* (marriage) → *menikah* (to get married) *rusak* (to break) → *merusak* (breaking) *wabah* (epidemic) → *mewabah* (outbreak) *yakin* (sure) → *meyakin(kan)* (to convince someone)
peng	peng	+ Vowel|k|g|h…, e.g: *ikat* (to tie) → *pengikat* (something that is used to tie)

Prefix	Variation Form	Rules
		olah (to process) → *pengolah* (processor) *ukur* (to measure) → *pengukur* (measurement) *urus* (to take care) → *pengurus* (person who take cares) *ganggu* (to disturb) → *pengganggu* (person who disturbs) *halus* (soft) → *penghalus* (softener)
	peny	+ s…, e.g: *saring* (to filter) → *penyaring* (filter)
	pem	+ b|f|p…, e.g: *baca* (to read) → *pembaca* (reader) *fitnah* (to accuse) → *pemfitnah* (people who accuse) *pukul*(to hit) → *pemukul* (things that is used to hit)
	pen	+ c|d|j|t… *cuci* (to wash) → *pencuci* (laundress/laundryman) *datang* (to come) → *pendatang* (the comer) *jual* (to sell) → *penjual* (seller) *tukar* (to change) → *penukar* (changer)
	pe	+ l|m|n|r|y|w…, e.g: *lintas* (to cross) → *pelintas* (passerby) *makan* (to eat) → *pemakan* (eater) *rusak* (to break) → *perusak* (destroyer) *warna* (color) → *pewarna* (dye)
ber	bel	+ ajar, eg: *ajar* (to teach) → *belajar* (to study/to learn)
	be	+ r|KVr…, e.g: *rencana* (plan) → *berencana* (to have a plan)
		kerja (to work) → *bekerja* (working)
	ber	+ any word which violates conditions of the alomorphs bel and be *tukar* (to change) → *bertukar* (to change, changing)
per	pel	+ ajar, e.g: *ajar* (to teach) → *pelajar* (student)
	pe	+ r|KVr…, e.g: *ramal* (to predict) → *peramal* (fortune-teller)
	per	+ any word which violates conditions of the alomorphs pel and pe *kaya* (rich) → *perkaya* (to make richer)
ter	te	+ r…, e.g: *rasa* (to feel) → *terasa* (to be felt)
	ter	+ K|V…, where K ≠ r, e.g: *atur* (to arrange) → *teratur* (to be properly arranged)

- The Meaning of Affixiation

TABLE III. MEANING OF AFFIXIATION [6]

Affix	Functions	Examples
meng-	verb to verb form noun to verb form	*makan* (to eat) → *memakan* (to eat, eating) *sisir* (comb) → *menyisir* (to comb)
di-	verb to passive verb form noun to passive verb form	*makan* → *dimakan* (to be eaten) *sisir* → *disisir* (to be combed)
ter-	verb to passive accidental verb form noun to passive accidental verb form	*makan* → *termakan* (to be eaten accidently) *paku* (nail) → *terpaku* (to get nailed accidently)
peng-	noun to noun form verb to noun form adjective to noun form	*tani* (farm) → *petani* (farmer) *baca* (to read) → *pembaca* (reader) *rusak* (damaged, destroyed) → *perusak* (destroyer)
ber-	verb to active verb form noun to active verb form adjective to active verb form	*main* (to play) → *bermain* (to play, playing) *sepeda* (bicycle) → *bersepeda* (to bike/cycling) *gembira* (happy) → *bergembira* (to be excited)
per-	verb to noun form noun to causative verb form adjective to causative verb form	*kerja* (to work) → *pekerja* (worker) *istri* (wife) → *peristri* (to take someone as a wife) *halus* (soft) → *perhalus* (to make softer)
ke-	adjective to noun form	*tua* (old) → *ketua* (leader)
-kan	verb to command verb form noun to command verb form adjective to command verb form	*ambil* (to take) → *ambilkan* (asking someone to take) *sisir* → *sisirkan* (asking someone to comb something) *jauh* (far) → *jauhkan* (asking someone to move something further)
-i	verb to intensive/repetitive verb form noun to intensive/repetitive verb form adjective to commmand verb form	*ambil* → *ambili* (taking something several times) *sisir* → *sisiri* (combing something several times) *jauh* → *jauhi* (asking someone to move further from something)
-an	verb to noun form	*makan* → *makanan* (food, something to be eaten)

Based on the above stemming rules, we get prefixes, suffixes and affixes to be used in the basic dictionary.

2) Symbol And Numeric Rules

The new codeword of symbol and numeric are stored in basic dictionary. The codeword of symbol and numeric is made of one or two number of characters. So that it is possible to separate the codeword of number and numeric.

3) Capital Letter Rule

To create a new codeword of a word containing one capital letter of first character of a word and capital letters of all characters of a word, a flag is used to mark the capital letter on a word.

The examples of the requirements are:

- Capital letter of first character of a word.
  The code word of word is created. For example: "Saya".
  First: The codeword of "saya" from dictionary is obtained, as an example is "**63**".
  Second: a flag "0" is added to the codeword"**63**" so that "Saya" becomes "0 **63**" .
- Capital letters of all characters of a word
  The code word of word is created, for example: "SAYA".
  First: The codeword of "saya" from dictionary is obtained, as an example is "**63**".
  Second: a flag "00" is added to the codeword "**63**" so that "SAYA" becomes "00 **63**".

Table IV shows the contribution in this study compared to the previous ones.

TABLE IV. OUR CONTRIBUTIONS

No.	Characteristics	WB-LZW	WB-Adaptive Huffman	Our Contributions
1	Extraction	Word→word, Symbol alphanumeric →Character	Word, symbols→Word	Word, symbols→word
2	Dictionaries	word and non-word	(7, 9, 11, 13) bits tables. Most frequently used words were placed in smallest bits table.	basic and supplement
3	Initialized Dictionary	-	9572 English words	symbols, 1-2 digit numbers, conjunctions, prefixes, suffixes, affixes, most frequently used Indonesian words (total 903 entries).
4	Searching / matching procedure	each entry on table	each entry on table	Trie (prefix tree) structure on table
5	Encoding	Bits	Bits	String
6	Capital characters	Different codes	Different codes	flags + stem words
7	Stemming	-	-	Indonesian Stemming
8	Compression ratio calculation	generated codes / source file	generated codes / source file	compressed file size / source file

III. EXPERIMENT AND ANALYSIS

On Table I, all the experiment was conducted on English language document. It was not conducted on Indonesian language document that contain many affixes, prefixes, and suffixes. We suppose that if all of the methods listed in table 1 were tested on documents written in Bahasa Indonesia, then the compression ratio will be less than the performances listed in Table I, since English words do not use affix, prefix and suffix and variation of embedding them on basic Indonesian word will introduce additional codewords in the dictionary, necessitating larger dictionary size and reducing less compression ratio.

Table V shows the experiment results of our method for some types of Indonesian language text file with different file sizes. In this experiment the file size of source file and the file size for the supplement table were recorded.

TABLE V. COMPRESSION OF COMPLEX FILE UNITS

Original Size in bytes	Compression Result Size	
	Bytes	%
11236	3612	32.15
20584	4219	32.17
30934	4675	31.26
40476	5270	31.41
51597	5802	31.09
103356	6378	30.98
204541	7095	30.98
303130	7685	30.89
403432	8226	30.57
503832	8827	30.56

Comparing with performances listed in Table I (in file sizes of 24K – 83K bytes), our results were comparable and in some cases outperform those listed in the table.

TABLE VI. COMPRESSION TIME COMPARISON BETWEEN THE PROPOSED TEXT COMPRESSION METHOD WITH REVERSED SEQUENCE OF CHARACTERS ON LZW [13]

Original Size in bytes	Compression Time (ms)	
	Reverse Sequence of Character LZW	Our proposed method
3977	31212	57
12305	101200	209
20821	181235	342
25422	243320	361
39175	342130	740

Since we didn't have information about time performance of the methods listed in Tabel I, we picked the work which has time performance. Table VI shows the comparison of compression time between our proposed method with reversed sequence of characters on LZW [13]. We can see that the compression time of our proposed method far outperform [13]. It is because the latter used linear search while ours adopted Trie as a search method. On linear search, data is searched from the beginning of the table to the end of the rows until the data is found, while on the Trie dictionary, data will be searched based on the initial character, and will continue to explore character by character in the nodes of tree until the data is found.

IV. CONCLUSION

Based on the above experiment on the file sizes of about 10K to -/+ 500K bytes and code-length of 16-bits, it was shown that the proposed method gave compression ratio of less than 50% of the original file size when viewed from the output file that consisting of a stream of codewords and supplement table.

The experiment can also show that the time required to compress/decompress file with the proposed method is far shorter than the method of Reversed Sequence of Characters on LZW [13].

It is acknowledged that this study still has some limitations and needs further study. For example, it is suggested that to increase the compression ratio, it is necessary to add more words in the basic dictionary, make a special treatment to handle words that most frequently appear by using the variable length code to handle those words.

REFERENCES

[1] R. Nigel Horspool and Gordon V. Cormack,"Constructing word-based text compression Algorithms", Data Compression Conference, pages 62–71, 1992.

[2] J. Jiang and S. Jones, "Word-based dynamic algorithms for data compression", Communications, Speech and Vision, volume 139, pages 582–586, December 1992.

[3] H. K. Reghbati, "An Overview of data compression techniques", Computer, Vol.14, No. 4, pp.71-76, July 1981.

[4] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression", IEEE Trans. Inf. Theory, vol. IT-23, no. 3, pp. 337–343, Mar.1977.

[5] Nazief, Bobby and Mirna Adriani, "Confix-Stripping: Approach to Stemming Algorithm for Bahasa Indonesia", Faculty of Computer Science University of Indonesia.

[6] Fadillah Z. Tala, "A Study of Stemming Effect on Information Retrieval in Bahasa Indonesia", Netherland, Universiteit van Amsterdam, http://ucrel.lancs.ac.uk/acl/P/P00/P00-1075.pdf, last accessed on 25 Juli 2009.

[7] David Salomon and Giovanni Motta, "Handbook of Data Compression", Fifth Edition, Springer-Verlag, London Limited, 2010

[8] Mengyi Pu, Ida, "Fundamental Data Compression", First Edition, London, Elsevier, 2006

[9] M. F. Porter, "An algorithm for suffix stripping", Program, 14(3):130–137, July 1980.

[10] Yusof. Mohd Kamir, Mat Deris, Mohd Sufian, Abidin, Ahmad Faisal Amri, Madi. Elissa Nadia, "Achieving Capability in Data Compression Using LZW++ Technique". International Journal of Computer Science and Network Security, Vol.9, No.8, August 2009.

[11] Maxime Crochemore and Thierry Lecroq, "Trie. Encyclopedia of Database Systems", 2009: 3179-3182.

[12] Devie Ryana Suchendra, "Improving Text Compression by Introducing Reversed Sequence of Characters on Lempel Ziv Welch (LZW) Algorithm", Telkom Institute Of Technology, 2012.