Khmer Word Segmentation based on Bi-Directional Maximal Matching for Plaintext and Microsoft Word Document

Narin Bi* and Nguonly Taing †
Royal University of Phnom Penh, Phnom Penh, Cambodia
narin.mite229@rupp.edu.kh* and taing.nguonly@rupp.edu.kh†

Abstract— **One of major key component in Khmer language processing is how to transform Khmer texts into series of separated Khmer words. But unlike in Latin languages such as English or French; Khmer language does not have any explicit word boundary delimiters such as blank space to separate between each word. Moreover, Khmer language has more complex structure to word form which causes Khmer Unicode standard ordering of character components to permit different orders that lead to the same visual representation; exactly looking word, but different character order. Even more, Khmer word could also be a join of two or more Khmer words together. All these complications address many challenges in Khmer word segmentation to determine word boundaries. Response to these challenges and try to improve level of accuracy and performance in Khmer word segmentation, this paper presents a study on Bi-directional Maximal Matching (BiMM) with Khmer Clusters, Khmer Unicode character order correction, corpus list optimization to reduce frequency of dictionary lookup and Khmer text manipulation tweaks. The study also focuses on how to implement Khmer word segmentation on both Khmer contents in Plaintext and Microsoft Word document. For Word document, the implementation is done on currently active Word document and also on file Word document. The study compares the implementation of Bi-directional Maximal Matching (BiMM) with Forward Maximal Matching (FMM) and Backward Maximal Matching (BMM) and also with similar algorithm from previous study. The result of study is 98.13% on accuracy with time spend of 2.581 seconds for Khmer contents of 1,110,809 characters which is about 160,000 of Khmer words.**

Keywords: Bi-directional Maximal Matching, Forward Maximal Matching, Backward Maximal Matching , Word Segmentation, Khmer Unicode, Khmer Cluster, Khmer Unicode

I. INTRODUCTION

Word segmentation is one of the most basic but crucial task in analysis of languages without word boundary delimiters. In Natural language processing (NLP) like information retrieval, machine translation, speech processing, etc.; words boundary is very important.

On basic, word segmentation based on two approaches: supervised and unsupervised. Particularly, supervised segmentation approach can be achieved a very high precision on the targeted knowledge domain with the help of training corpus, the manually segmented text collection. On the other hand, unsupervised segmentation approach where there is no or limited training data available. The resulting segmentation accuracy with unsupervised approach may not be very satisfying, but the effort of creating the training data set is not absolutely required.

Bi-directional Maximal Matching (BiMM), the supervised segmentation approach, is fundamentally the combination of Forward Maximal Matching (FMM) and Backward Maximal Matching (BMM) algorithm which solve ambiguity problems caused by greedy characteristic of Maximal Matching algorithm when using alone; either FMM or BMM.

II. RELATED WORK

There are many works related to text segmentation are proposed with many new techniques based on supervised and unsupervised segmentation approach for languages like Chinese, Japanese, Thai, and Vietnamese.

For Khmer word segmentation, the paper on Khmer Unicode Text Segmentation using Maximal Matching [2], use Maximal Matching algorithm combined with Khmer Cluster, Khmer Unicode Sorting technique based on decimal code, and hash table for dictionary look-up. With these techniques, Khmer text segmentation could achieve up to 97% segmentation accuracy and using 0.2321 second; where the result comes from the experiment of 10 random documents from newspapers.

The paper on Word Bigram vs. Orthographic Syllable Bigram in Khmer Word Segmentation [7], discusses the word segmentation of Khmer written text based on Bigram model. The research carries out two extended methods from the Bigram Model, Word Bigram and Orthographic syllable Bigram. The sound similarity errors identification of their previous research are combined to improve the accuracy of the segmentation which based on the maximum matching algorithm for the word segmentation. Many texts with different genres were collected and manually segmented. The size of the training corpus is 10.6 MB with 673295 words and 20991 vocabularies. Some random texts were selected from the newspaper and the book for evaluating the accuracy of approach. The text is about 190 KB in size with 13025 words. For experimentation, they compare the automatic segmented text with the manual segmented text. According to the experimentation, Word Bigram outperforms the Orthographic syllable Bigram approach (see Table I).

TABLE I
WORD BIGRAM VS KCC BIGRAM RESULT

	Word Bigram	KCC Bigram
No. of words	12760	11602
Precision Rate	91.562	72.327
Recall Rate	92.138	72.438
F-measure	91.849	72.382

Another research paper on Development of a Khmer Spell Checker Based on a Hidden Markov Model [3], Khmer words are segmented based on the combination of a hidden Markov model and a dictionary look-up technique. With this strategy, the spell checker could achieve up to 95% segmentation accuracy for documents with a small number of misspelled words. The accuracy drops in documents with a high number of misspelled words due to the fact that the hidden Markov model was not trained to segment misspelled words.

III. KHMER LANGUAGES

Khmer script is the official language of Cambodia. Khmer script adapted from the Pallava script, which was used in southern India and south East Asia during the 5th and 6th Centuries AD. The oldest dated of inscription in Khmer was found at Angkor Borei in Takeo Province south of Phnom Penh and dated from 611 AD [1].

Khmer script is written from left to right with multiple levels of character stacking possible and also uses diacritics that further enhance the pronunciation of words. Originally, there are 35 consonants (see Table II), but only 33 are now in use. The vowel system consists of 14 independent vowels (see Table III) and 23 dependent vowels (see Table IV); alongside there are also 10 digits (see Table V), 2 consonant shifters, some diacritic signs and special characters (see Table VI).

TABLE II
KHMER CONSONANTS AND CONSONANT SUBSCRIPTS

ក [kâ] ◌	ខ [khâ] ◌	គ [kô] ◌	ឃ [khô] ◌	ង [ngô] ◌
ច [châ] ◌	ឆ [chhâ] ◌	ជ [chô] ◌	ឈ [chhô] ◌	ញ [nhô] ◌(៣)
ដ [dâ] ◌	ឋ [thâ] ◌	ឌ [dô] ◌	ឍ [thô] ◌	ណ [nâ] ◌
ត [tâ] ◌	ថ [thâ] ◌	ទ [tô] ◌	ធ [thô] ◌	ន [nô] ◌
ប [bâ] ◌	ផ [phâ] ◌	ព [pô] ◌	ភ [phô] ◌	ម [mô] ◌
យ [yô] ◌	រ [rô] ◌	ល [lô] ◌	វ [vô] ◌	ឥ * [shâ] ◌
ឣ * [ssô] ◌	ស [sâ] ◌	ហ [hâ] ◌	ឡ [lâ]	អ [qâ] ◌

* These consonants are not use in modern Khmer language

TABLE III
KHMER DEPENDENT VOWELS

◌ា [a] / [éa]	◌ិ [ĕ] / [ĭ]	◌ី [ei] / [i]	◌ឹ [œ̆]	◌ឺ [œ]	◌ុ [ŏ] / [ŭ]	◌ូ [o] / [u]	◌ួ [uŏ]
◌ើ [aeu]/[eu]	◌ឿ [eua]	◌ៀ [iĕ]	◌េ [é]	◌ែ [ê]	◌ៃ [ai]/[ey]	◌ោ [aô]/[oŭ]	◌ៅ [au]/[ŏu]
◌ុំ [om]/[ŭm]	◌ំ [âm]/[um]	◌ាំ [ăm]/[ŏám]	◌ះ [ăh]/[eăh]	◌ុះ [ŏh]/[uh]	◌ោះ [éh]	◌ោះ [aôh]/[uŏh]	

TABLE IV
KHMER INDEPENDENT VOWELS

ឥ [ĕ]	ឦ [ei]	ឧ [ŏ]	ឩ *	ឩ [ŭ]	ឪ [ŏu]	ឫ [rœ̆]	ឬ [rœ]
ឭ [lœ̆]	ឮ [lœ]	ឯ [é]	ឰ [ai]	ឱ , ឲ [aô] , [aôy]	ឳ [âu]		

* These Independent Vowels is not use in modern Khmer language

TABLE V
KHMER NUMERALS

Khmer Numerals	០	១	២	៣	៤	៥	៦	៧	៨	៩
Arabic Numerals	0	1	2	3	4	5	6	7	8	9

TABLE VI
KHMER DIACRITIC SIGNS AND OTHER SPECIAL CHARACTERS

◌ំ	◌ះ	◌ៈ	◌៉	◌៊	◌ ◌	◌ ◌	◌ ◌
◌ ◌	◌ ◌	◌ ◌	◌ ◌	◌ ◌	◌ ◌	៎	៏
។	៕	៖	?	!	,	.	"
«	»	◎					

IV. MATERIALS AND METHODS

A. *String Object*

StringBuffer and **StringBuilder:** are mutable string, unlike regular strings which are immutable. We can modify string as many times as necessary without recreating String object. Anyhow, StringBuffer and StringBuilder may have greater overhead for one String, but much less overhead for the many Strings that would be necessary to do what could be done with one StringBuffer or StringBuilder; especially in the situation that we need to concatenate many strings together that produces surprising performance improvements.

String interning: based on flyweight design pattern, it is a technique of storing only one original copy of distinct string value in a string intern pool which must be immutable. String interning could speed up string comparisons that rely heavily on hash tables with string keys. Without interning, checking two equal different strings are involves examining every character of both strings which is slow.

B. Khmer Clusters

Each Khmer Cluster contains one based character which includes consonants, independent vowels, numbers and some other signs. A cluster can have up to two subscripts, two vowels, a ROBAT (‘◌ំ’), a consonant shifter (‘◌៎’ or ‘◌៏’) or one of these signs (‘◌៉’, ‘◌៊’, ‘◌់’, ‘◌៌’ and ‘◌៑’). Khmer words are the combination of one or more consonantal clusters [3].

CLUSTER: = B {R|C} {S{R}}* {{Z}V} {O} {S}

Where B is a base character, R is a ROBAT, C is a consonant shifter, S is a consonant subscript, Z is an invisible space or a zero width non-joiner, V is a vowel and O is any other sign.

TABLE VII
EXAMPLE OF CLUSTER

Words	No. Clusters	Clusters
ស្ត្រី	1	[ស្ត្រី]
សូស្តី	2	[សូ] + [ស្តី]
ចម្រៀង	3	[ច] + [ម្រៀ] + [ង]
បរិយាកាស	5	[ប] + [រិ] + [យា] + [កា] + [ស]

The input text may consist of other characters which are not in range of Khmer Unicode characters, so there are some conditions added to handle with those characters. Non-Khmer Unicode remains intact and stores as a cluster. There is also a condition to handle with character signs (‘,’ and ‘.’) which appear between Khmer numerical characters, so Khmer numerical characters and those character signs stored as one cluster. Special case for a space character (‘ ‘) or group of space characters (‘ ‘) are stored as one cluster.

“លោក Smith” = [លោ] + [ក] + [ ] + [Smith]

In another research paper, this similar algorithm could be referring to as Khmer Character Cluster (KCC) which is a combination of characters by mean inseparable unit in writing. Khmer word can be the combination of one or more KCCs. Text is segmented to KCCs by detecting the transition state for each character [5].

C. Khmer Unicode Spelling Order

Ordering of Khmer word spelling [6] is as follow:
1st: Base consonant
2nd: First subscript
3rd: Second subscript
4th: Consonant shifter
5th: Vowel
6th: Various signs

Khmer consonant subscripts are divided into 3 categories and the priorities of the input sequences are:
1st: South subscript (placed below of base character)
2nd: East subscript (place on right side of base character)
3rd: West subscript (place on left side of base character)

Khmer vowels are divided into 4 categories. The priorities of the input sequences are:
1st: North vowel or East vowel,
2nd: West vowel and
3rd: South vowel.

For independent vowel and digit are considered the stand-alone character.

D. Khmer Unicode Character Order Correction

Khmer Unicode order of character components based on rule of Khmer word spelling orders; but still permits different character orders that lead to the same graphical representation (the same looking word, but different character order). Even more, there are some forms of writing word which even not looking exactly the same but closely similar to each other.

The combination of these issues could lead to many errors when doing dictionary look up to segment Khmer words; although those words are existed in corpus list, the characters ordering of these words are differences.

Example: Word “ស្ត្រី”

(1) ស + ◌្ + រ + ◌្ + ត + ◌ី = ស្ត្រី
(2) ស + ◌្ + ត + ◌្ + រ + ◌ី = ស្ត្រី
(3) ស + ◌្ + ត + ◌ី + ◌្ + រ = ស្ត្រី
(4) ស + ◌្ + រ + ◌ី + ◌្ + ត = ស្ត្រី

(1), (2) and (3) are formed the exact same word “ស្ត្រី”, even each word has different characters order. For (4), it looks very similar to (1), (2) and (3).

TABLE VIII
ALL POSSIBLE INTERCHANGEABLE POSITION OF CHARACTER TYPES

1st Position After Main Character		**2nd Position After Main Character**
SSS, ESS, WSS	⇔	Consonant Shifter
WSS	⇔	SSS, ESS
WV, EV, NV, SV	⇔	WSS
WV, EV, NV	⇔	SSS
WV, NV	⇔	ESS
Various Sign	⇔	SSS, ESS, WSS, EV, Vowel (‘◌ោ’ and ‘◌ៅ’)

SSS: SOUTH SUBSCRIPT, **ESS**: EAST SUBSCRIPT,
WSS: WEST SUBSCRIPT, **WV**: WEST VOWEL, **EV**: EAST VOWEL,
NV: NORTH VOWEL, **SV**: SOUTH VOWEL

Because there are issues as mention above that need to be solved. We have constructed a list of all possible interchangeable position of character types that could form the same word or very similar word as showed in Table VIII.

The process of re-order characters position proposed in this study is implemented on Khmer cluster, not on Khmer word or sentence; Because Khmer cluster is well defined and consists of only one main character in each cluster.

The changing position of character is based on location of each character compare to main character. Mainly only characters in first and second position after main character are needed to interchange each other. Nevertheless, there is an exceptional case for West subscript (WSS) which could appear on position beside first and second after main character that needs to be re-ordered the position. All shifting position is based on ordering priority of Khmer script in Khmer Unicode Spelling Order.

Problem words:

(1) ស + ◌្ + រ + ◌្ + ត + ◌ី = ស្ត្រី

(2) ស + ◌្ + ត + ◌្ + រ + ◌ី = ស្ត្រី

(3) ស + ◌្ + ត + ◌ី + ◌្ + រ = ស្ត្រី

(4) ស + ◌្ + រ + ◌ី + ◌្ + ត = ស្ត្រី

All confusing ordering words are re-ordered the characters position into a correct single word form:

ស + ◌្ + ត + ◌្ + រ + ◌ី = ស្ត្រី

E. Corpus list

The collection of all possible Khmer words is stored in a Corpus list (Word list). We use this list of corpus for dictionary lookup to identify word during the process of word segmentation.

There are two groups of data in the corpus list: word and its indicated number. The indicated number of each word identify whether there is another word in this list that begin or end with this word.

All words store in corpus list are based on these structures:

- There is no duplicate word in the list
- Words are not required to be in alphabet order
- Each word must assigned an indicated number (0,1,2 or 3)
- Each word separated by new line (“\n”)
- Word and its indicated number separated by tab (“\t”)

To optimize dictionary lookup on corpus list, we assign an indicative number to each word to reduce the number of times need to do look up for word in corpus list. There is no use to look for word which does not exist in the corpus list.

To make sure that the indicative number is surely beneficial of improving the performance of word segmentation, we have done the analysis on all words in the corpus list. We found out that only small amount of words have another word(s) as the beginning or ending. In total of around 84,409 words in the collection of our word list that use for experimentation, there are only 6,980 words (8.27%) have word(s) that began and ended with another words (see Table IX).

TABLE IX
LIST OF INDICATED NUMBER FOR WORDS IN CORPUS LIST

Indicated No.	Description	Number of Words in Corpus List (%)	Example
0	There are word(s) that began and ended with this word	6,980 Words (8.27%)	និយម 0
1	There is no word that ended with this word	6,700 Words (7.94%)	អនិយម 1
2	There is no word that began with this word	7,891 Words (9.35%)	ជ្រុលនិយម 2
3	There is no word that began or ended with this word	62,838 Words (74.44%)	និយមន័យ 3

To reduce the errors when lookup for word in the corpus list, we have implemented method to correct character order of each word based on Khmer word spelling rules as mentioned above and also mapped consonant subscript “TA” and “DA” (◌្ត) to the same character code when generated corpus list.

The main purpose is to prevent the mistake of inputting the word which has many way of writing and different word that look the same in the corpus list.

Example 1: Word “ស្ត្រី”

ស + ◌្ + រ + ◌្ + ត + ◌ី = ស្ត្រី

ស + ◌្ + ត + ◌្ + រ + ◌ី = ស្ត្រី

Example 2: Subscript of consonant: “ដ” and “ត”

◌្ + ដ = ◌្ដ

◌្ + ត = ◌្ត

F. Bi-directional Maximal Matching for Khmer Word Segmentation

Forward Maximal Matching (FMM) is the dictionary-based algorithm. It is frequently referred to as greedy algorithm. This algorithm starts from the beginning of a text and based on corpus list (Word list) for segmented word. The algorithm tries to find the longest word. If a word is found, the Forward Maximal Matching algorithm marks a boundary at the end of the longest word, and then begins the same

longest match search starting at the character following the match. If there is no word found in the corpus list, the algorithm keeps that character and begins search starting at a next character. Forward Maximal Matching algorithm for Khmer Word Segmentation in this study implemented with Khmer Clusters. So instead of loop through character by character, it loops through list of Khmer Clusters.

Backward Maximal Matching (BMM) is quite similar to Forward Maximal Matching. The only difference is on the direction of scanning text. This algorithm starts from the end of a text to the beginning. The algorithm tried to find the longest word. If a word is found, it marks a boundary at the start point of that word, and then begins the same longest match search starting at the character in front of the match. If there is no word found in the corpus list, the algorithm keeps that character and begins search starting at a character in front of it.

Bi-directional Maximal Matching (BiMM) is the combination of FMM and BMM. The mechanism behind both algorithms are quite similar, the difference is on the direction of scanning text. While FMM scan and segment text from the beginning to the end, BMM is start from the end to the beginning of text backward. The advantage of using Bi-directional Maximal Matching is to solve the problem of ambiguity in Khmer Word Segmentation which could not be solved when using FMM or BMM algorithm only; because of the characteristic of both algorithms (see Table X).

TABLE X
AMBIGUITY IN FORWARD AND BACKWARD MAXIMAL MATCHING COMPARE TO BI-DIRECTIONAL MAXIMAL MATCHING

Forward Maximal Matching (FMM)
មនុស្សម្នាក់
=> មនុស្សម្នា + ក់ (Incorrect)
ប្រជាជនកម្ពុជាមួយពាន់នាក់
=> ប្រជាជាន + កម្ពុជា + មួយ + ពាន់ + នាក់
Backward Maximal Matching (BMM)
មនុស្សម្នាក់
=> មនុស្ស + ម្នាក់
ប្រជាជនកម្ពុជាមួយពាន់នាក់
=> ប្រជាជាន + កម្ពុ + ជាមួយ + ពាន់ + នាក់ (Incorrect)
Bi-directional Maximal Matching (BiMM)
មនុស្សម្នាក់
=> មនុស្ស + ម្នាក់
ប្រជាជនកម្ពុជាមួយពាន់នាក់
=> ប្រជាជាន + កម្ពុជា + មួយ + ពាន់ + នាក់

The implementation processes of Bi-directional Maximal Matching for Khmer Word Segmentation are as follow:

- First, preload Corpus List into Hash Table (Generic Dictionary Object) and then generate Khmer Clusters from inputted unsegment text and loop through list of those Khmer Clusters.
- List of Khmer Clusters is divided into sub-groups based on the first character of each cluster during looping. If the beginning character of cluster is not a Khmer consonant, a Khmer number, or a Latin character, sub-group is defined.
- Next, generate list of word segmentation from sub-groups, both FMM and BMM and store in each temporary List. The maximum length of Khmer word from FMM and BMM is not more than 12 clusters in length.
- Choose List with minimum number of segmentation and unknown word count to add to the final segment List.
- Finally, return the final List of segmented words

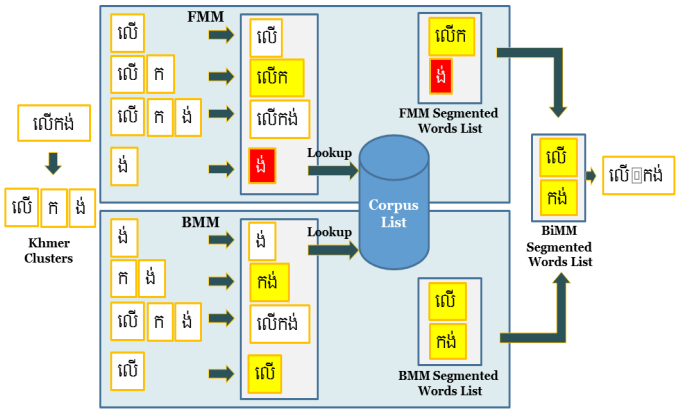

Fig. 1 Process Flow of Bi-directional Maximal Matching

V. IMPLEMENTATION ON PLAINTEXT AND WORD DOCUMENT

The study of Khmer word segmentation in this paper is implemented on Khmer Unicode contents in three contexts: Plaintext (*.txt), Microsoft Word document (*.docx) and current opened Word document for experimentation.

A. Plaintext

The process is separated into four parts: Pre-processing that is done before segmentation, Semi-segmentation, Process Word Segmentation, and Post-processing after segmentation.

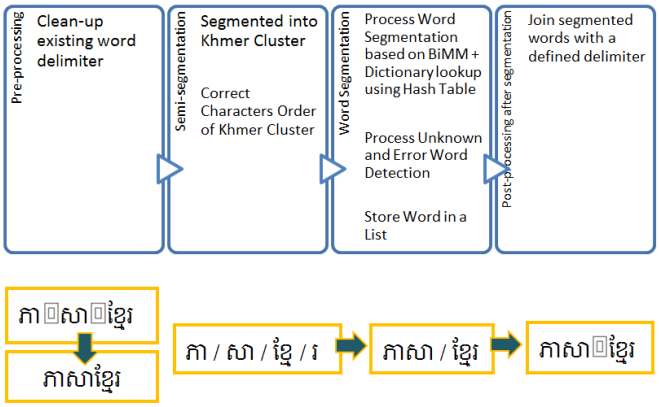

Fig. 2 Process Flow of Khmer Word Segmentation for Plaintext

Pre-processing: Before process on word segmentation, we do clean-up on all existing word delimiter (“\u200B”) to prevent duplication and prepare an unsegment contents.

Semi-segmentation: Based on the concept of Khmer Cluster [3], and Khmer Unicode spelling order [8], document is segmented into Khmer Clusters, correcting the characters order in each cluster and store in List for joining process.

Word Segmentation: Khmer Clusters are joined together one by one for comparing and the consonant subscript “TA” and “DA”, which look the same (◌ྟ), are mapped to the same character code (“\u17D2\u178A”). Then the combinations of clusters are used for dictionary lookup based on Bi-directional Maximal Matching Algorithm. Found words are stored in segmented word list and the unknown/error word are process through Unknown and Error Word Detection Method and also store in segmented word list.

Post-processing after segmentation: The final step is to join the segmented word get from previous steps with predefined Khmer word delimiter to form the final segmented document. The joining process is based on delimiter joining rule.

B. Word Document

The implementation of Khmer Word Segmentation for Word document is based on OpenXML SDK, released by Microsoft [14], to manipulate the content; because all Word document from version 2007 and above (*.docx) are based on OpenXML format, so by mean all contents are the XML files.

The processes are separated into three parts: Pre-processing, Word segmentation, and Post-processing after segmentation.

Pre-processing: Based on OpenXML SDK, extract text content of Main Documents, Header, Footer, Endnotes and Footnotes of OpenXML from Microsoft Word document; then combine all texts in each paragraph and clean-up existing word delimiter.

Word segmentation: Process word segmentation with Plaintext.

Post-processing after segmentation: Join all segmented word with predefined Khmer word delimiter to form a new paragraph and new OpenXML Content and save those newly formed document to new Word document.

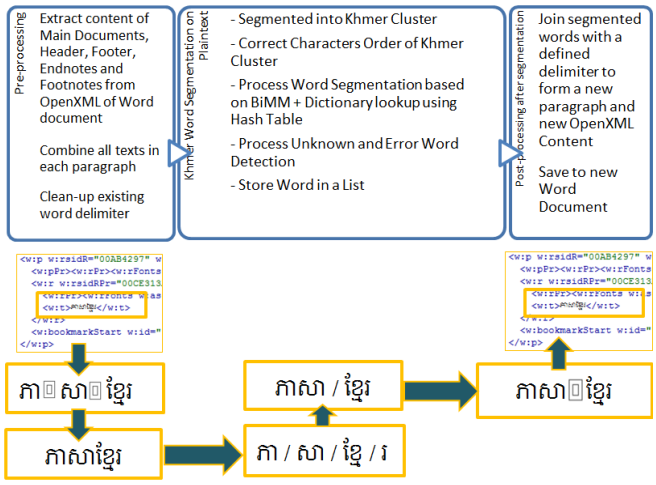

Fig. 3 Process Flow of Khmer Word Segmentation for Word Document

C. Current Opened Word Document

The process are similar to file Word document, but because of OpenXML SDK is not work natively with current opened Word document, so there are more steps need to be done before and after the content manipulated by OpenXML SDK [14].

Similarly, the processes are separated into three parts: Pre-processing, Word segmentation, and Post-processing after segmentation.

Pre-processing: Get word range of Main Documents, Header, Footer, Endnotes, and Footnotes. Exceptionally, the word range of Main Documents are separated into many sub-sections based on Page Break; because during the experimentation we have found out that the extra processes which allow current active Word document to work with OpenXML SDK could not handle with large size document. Then prepare the Flat OPC XML [16]. for each content part that we get from each Word range, Flat OPC XML is the technique which allows OpenXML SDK to handle with current active Word document. Next, prepare an in-memory package from each Flat OPC XML and get OpenXML content of each content part. Combine all texts in each paragraph and clean-up existing word delimiter.

Word segmentation: Process word segmentation with Plaintext.

Post-processing after segmentation: Join all segmented word with predefined Khmer word delimiter to form a new paragraph and new OpenXML Content. Lastly, convert back to Flat OPC XML from each in-memory package and inserts Flat OPC XML content back to identified word range.

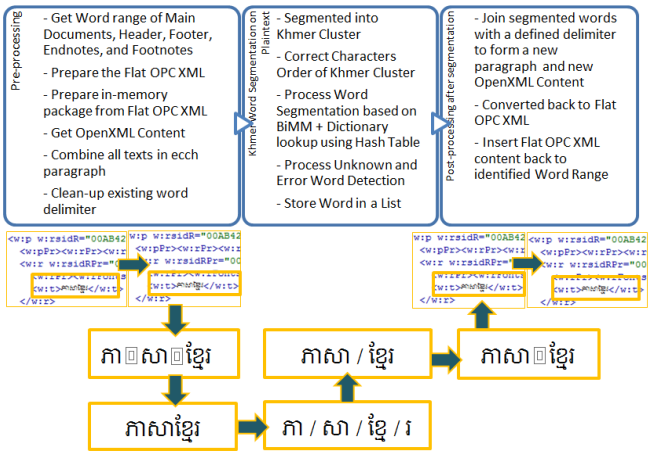

Fig. 4 Process Flow of Khmer Word Segmentation for Current Opened Word Document

VI. EXPERIMENTATION

The experimentation is divided into three groups: Plaintext (*.txt), Microsoft Word document – focus on Word document with extension (*.docx) and current opened Word document.

The experimental results are compared with previous research such as PAN Text Line Breaker [1] and Khmer Unicode Text Segmentation using Maximal Matching [2]. on Plaintext. For Word document, the comparison is with PAN Khmer Line Breaker[2] (Microsoft Word Add-in). Additionally, compare with FMM and BMM.

There are about 85,000 words in the corpus List for doing testing. Document for experiment are randomly selected from general administration letter, books, and Khmer news website.

A. Result of Accuracy

The level of accuracy is based on the number of found words compare to total number of found word and error word.

Accuracy = (N/T) * 100
N: Number of Found Words
T: Number of Found Words and Error word

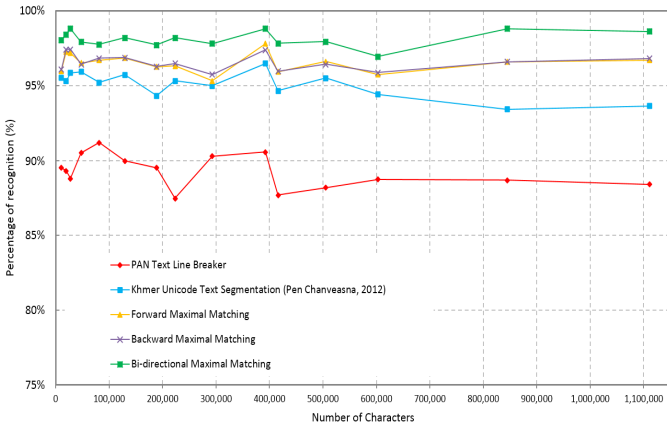

Fig. 5 Accuracy Evaluation

The graph in Fig. 5 shows that BiMM giving the highest level of accuracy. FMM and BMM are closely together, and over Khmer Unicode Text Segmentation using Maximal Matching [2]. PAN Text Line Breaker giving the lowest level among all.

B. Result of Time Consuming

Each algorithm has integrated a timer to trace the execute time. Only PAN Text Line Breaker and PAN Line Breaking have to manual record the duration of time execution as we do not have the source code of the software, so we minus 1.5 seconds from the duration when traced. The processing duration of each algorithm is in second.

Performance Testing are separated into two: Plaintext and Word Document.

Plaintext: consists of PAN Text Line Breaker, Khmer Unicode Text Segmentation [2], A modified version of Khmer Unicode Text Segmentation from [2][3], FMM, BMM, and BiMM.

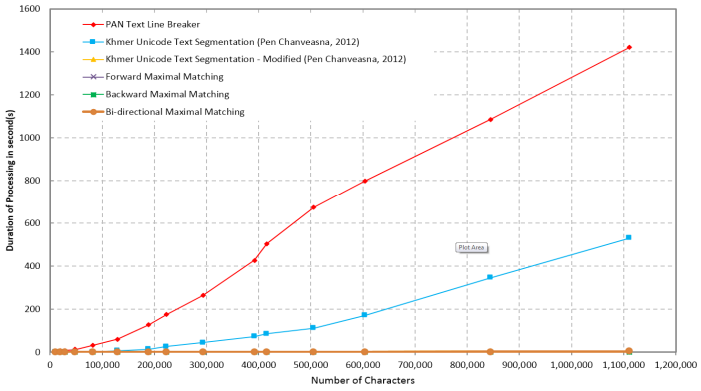

Fig. 6 Time Consuming of PAN Text Line Breaker and Khmer Unicode Text Segmentation (Plaintext)

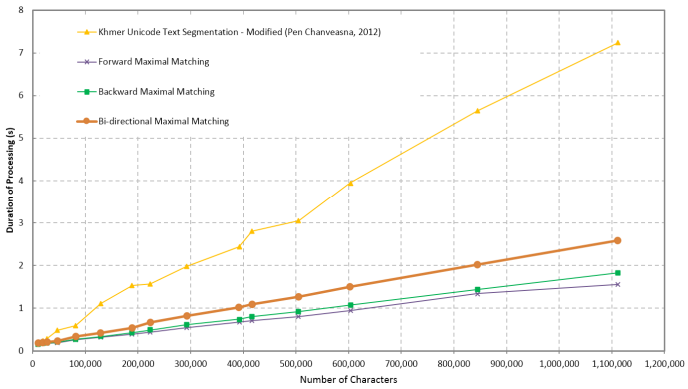

Fig. 7 Time Consuming of Khmer Unicode Text Segmentation – Modified, FMM, BMM & BiMM (Plaintext)

Based on graph in Fig. 6 and Fig. 7, FMM and BMM are quite similar in speed and the fastest among all. BiMM is just a little bit slower compare to FMM and BMM.

PAN Text Line Breaker is the slowest. Khmer Unicode Text Segmentation [2] is fast for small text content, but the performance drop when the size of content increased. The modified version of Khmer Unicode Text Segmentation has showed a better performance than the original version, but slower than FMM, BMM and BiMM.

[1] PAN Text Line Breaker is tool developed by PAN Localization Team for breaking Khmer contents to works separated by a delimiter.

[2] PAN Khmer Line Breaker has the same functionality as PAN Text Line Breaker, but running in Microsoft Word.

[3] A modified version of Khmer Unicode Text Segmentation has a modification on join string method of segmented word list from the original algorithm by Chanveasna Pen [2]. There is no other changed beside that.

Word document: consist of PAN Line Breaking, FMM, BMM, and BiMM.

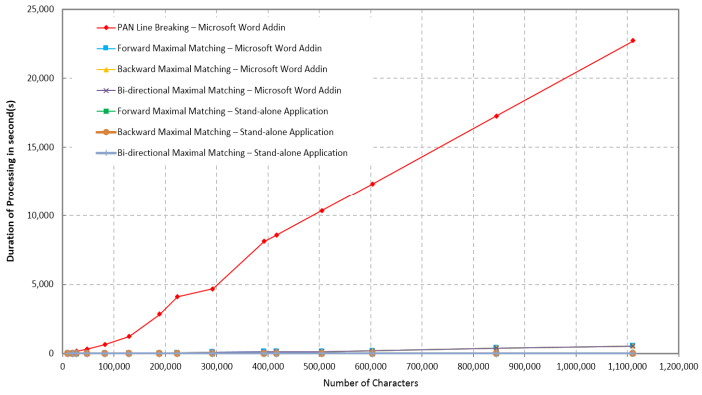

Fig. 8 Time Consuming of PAN Line Breaking (Word Document)

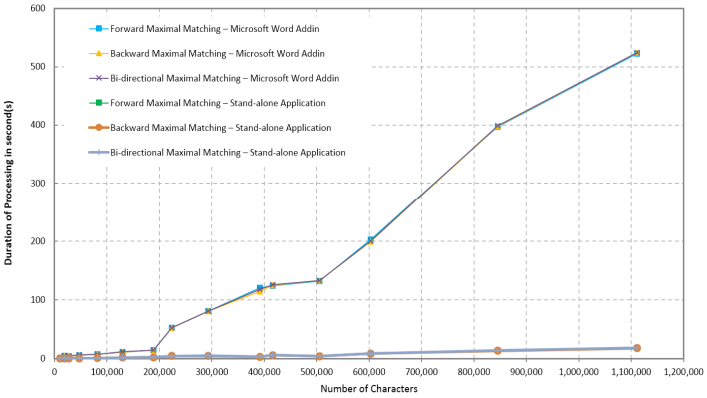

Fig. 9 Time Consuming of FMM, BMM & BiMM
both MS. Word Add-in & Stand-alone App. (Word Document)

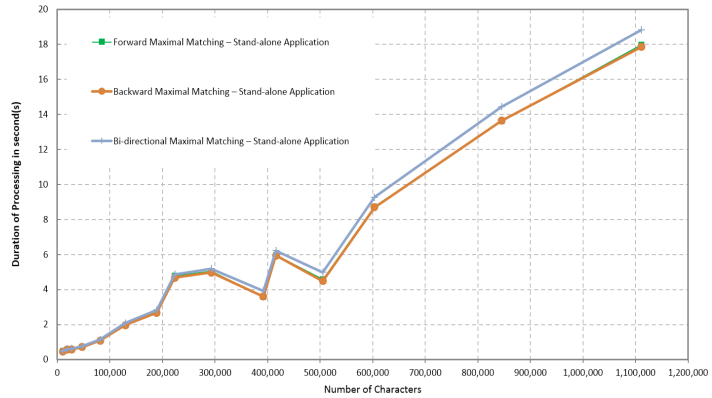

Fig. 10 Time Consuming of FMM, BMM & BiMM
of Stand-alone App. (Word Document)

Based on graph in Fig. 8, Fig. 9 and Fig. 10, the algorithm in stand-alone application (Word document file) is faster than the same algorithms implemented on current opened Word document. BMM and FMM still the fastest and follow by BiMM. The characteristic of algorithm is the same compare between Plaintext content and Word document.

VII. Discussion

A. *Accuracy*

BiMM gives an average accuracy of 98.13% and has higher accuracy rate than FMM and BMM because the mechanism behind BiMM is the complement of FMM and BMM together. FMM has higher accuracy rate than Khmer Unicode Text Segmentation [2]. even both of them based on the same core algorithm; as we have optimized and add some more filters to increase level of accuracy. PAN Text Line Breaker get lower rate compare to others because of words in corpus list is limit.

B. *Time consuming*

FMM and BMM are among the fastest algorithm. The performance between two algorithms is quite similar. Closest to these algorithms is BiMM. While Forward and Backward Maximal Matching using 0.157s and 0.161s to process word segmentation on plaintext documents which contain 10,356 characters; And 1.554s and 1.827s for document which contain 1,110,809 characters (about 160,000 Khmer words), BiMM is using time of 0.173s and 2.581s that is quite close to each other. With the same situation, PAN Text Line Breaker is using 3.708s and 1,421.792s (around 23 minutes and a haft).

For Word document, FMM and BMM stills lead on performance and follow by BiMM.

Word document process segmentation on file Word document is much faster than in currently opened Word document in Microsoft Word.

C. *Challenges*

Even many challenges have been solved but still there are some more existed that need to be investigate and study more. Here are some that not solved by the proposed algorithm yet:

Misspelling Words and Unknown Words: as the proposed algorithm is the dictionary based algorithm, so the level of accuracy is mainly depended on the quality and number of word in the corpus list, so if word doesn't found in the corpus list because of missing typing or word doesn't exist, the result of segmentation maybe wrong.

Example: កាន់កាប = កាន់ + កាប
បដិវាគមេតិ = បដិ + វា + គ + មេ + តិ

Context sensitive word segmentation: there are some words have many difference forms of segmentation based on the context meaning and the proposed algorithm could not give the correct output of segmentation; as the correct segmentation based on the context of paragraph surrounding these words.

Example: លើកដប = លើក + ដប (Correct)
= លើ + ក + ដប (Correct)

VIII. Conclusion and Future Work

This paper has presented the use of Bi-directional Maximal Matching in Khmer Word Segmentation to solve ambiguity problems caused by greedy characteristic of Maximal Matching Algorithm. The algorithm shows how scanning direction technique in Maximal Matching, known as Forward Maximal Matching and Backward Maximal Matching help identify miss segmentation words. Moreover, many other techniques to help improve the accuracy in segmentation like improving rules when generating Khmer clusters and fixing the character order in Khmer clusters, which could lead to many faulty when doing dictionary lookup because of word have many form of writing.

In term of performance, there is a noticeable improvement too; because of tweak on string manipulation using StringBuilder and String Interning. StringBuilder make string concatenation much faster when there are a lot of string needs to join together, also reduce memory usage and garbage collector process. String Interning use string pool to store unique string. It reduces memory usage and increase speed of string search in string pool; but there are some drawback for string intern when there are a lot of string manipulation. Beside these, corpus list store in hash table also optimize to reduce number of times to do lookup. Each word in the list has an indicated number to which word could be an end or beginning of another word. The study also applied Khmer word segmentation on contents on both Plaintext and Word document.

Notice on experimentation, the result shows a positive sign on both performance and accuracy for both plaintext and Word document. The result of accuracy is 98.13% for algorithm Bi-directional Maximal Matching. On Plaintext document that content about 1,110,809 characters (around 160,000 Khmer words), Forward Maximal Matching is 1.554s, Backward Maximal Matching is 1.827s and Bi-directional Maximal Matching is 2.581s.

More future work, we need to increase and improve quality of Khmer corpus list, focus more on context sensitive word segmentation and how to detect misspelling words and unknown word. Furthermore, integrate with unsupervised segmentation approach like Bi-gram and any other algorithm.

References

[1] Wikipedia, Khmer Language.
Available at http://en.wikipedia.org/wiki/Khmer_Language

[2] Pen Chanveasna, “Khmer Unicode Text Segmentation Using Maximal Matching”, Master Thesis, Royal University of Phnom Penh, November 2012.

[3] Puthick Hok, “Development of a Khmer Spell Checker Based on a Hidden Markov Model”, Master Thesis, Department of Compute Science, Australian National University, November 2005.

[4] Darayong Tith, “A Study on Distributed Algorithms on Khmer Word Counting in Hadoop Environment for Large Corpus”, Master Thesis, Royal University of Phnom Penh, December 2011.

[5] Chea Sok Huor, Top Rithy, Ros Pich Hemy and Vann Navy, “Detection and Correction of Homophonous Error Word for Khmer Language”, PAN Localization Team, Cambodia, Working Papers 2004-2007.

[6] Chea Sok Huor, Atif Gulzar, Ros Pich Hemy, Neak Longchrea, “Encoding Conversion Utility for Khmer”, PAN Localization Team, Cambodia, Working Papers 2004-2007.

[7] Chea Sok Huor, Top Rithy, Ros Pich Hemy, Vann Navy, Chin Chanthirith and Chhoeun Tola, “Word Bigram Vs Orthographic Syllable Bigram in Khmer Word”, PAN Localization Team, Cambodia, 2007.

[8] Chea Sok Huor, Atif Gulzar, Ros Pich Hemy and Vann Navy, “Khmer Collation Development”, PAN Localization Team, Cambodia, Working Papers 2004-2007.

[9] Pieqian Liu, Jingjing Feng , Jiyu An, “An Improvement Method for The Ambiguous Fragments Discovery in Chinese Word Segmentation”, July 2010.

[10] Wirote Aroonmanakun, “Collocation and Thai Word Segmentation”, 2002.

[11] Nadir Durrani, Sarmad Hussain, “Urdu Word Segmentation”, June 2010.

[12] Chunyu Kit, Haihua Pan, Hongbiao Chen, "Learning Case-based Knowledge for Disambiguating Chinese Word Segementation: A preliminary study", 2002

[13] Javier Solá, “Issues in Khmer Unicode 4.0”, Open Forum of Cambodia, 2004

[14] Open XML SDK 2.0 for Microsoft Office
Available at: http://msdn.microsoft.com/en-us/library/office/bb448854%28v=office.14%29.aspx

[15] Office–Dev Center. “Increasing Word Automation Performance for Large Amounts of Data by Using Open XML SDK.”
Available at: http://msdn.microsoft.com/en-us/library/office/ff191178%28v=office.14%29.aspx

[16] Eric White, The Flat OPC Format.
Available at: http://blogs.msdn.com/b/ericwhite/archive/2008/09/29/the-flat-opc-format.aspx

[17] Java SE 6 Documentation. StringBuffer.
Available at: http://docs.oracle.com/javase/6/docs/api/java/lang/StringBuffer.html

[18] Wikipedia, String interning,
Available at: http://en.wikipedia.org/wiki/String_interning

[19] NetOffice - MS Office in .NET
Available at: https://netoffice.codeplex.com/releases/view/70943

[20] The Unicode Standard, “Khmer range”
Available at: http://www.unicode.org/charts/PDF/U1780.pdf