Paraphrase Detection for Thai Textual Documents

Patthamanan Isaranontakul, Thiraphat Meesumrarn* and Danuwat Isaranontakul

Computer and Information Technology Program, Faculty of Science and Technology,

Nakhon SawanRajabhat University, 60000

*Email: thiraphat.meesumrarn@gmail.com

Abstract

The phenomenon of plagiarism in the digital contents is raising in present day and become the huge problem for idea stealing detection. Fortunately, the automatic identification of sentences in textual documents plays an important role fighting to the idea piracy. In this paper, the developed system presents the methodology to deal with the paraphrase detection through Thai academic documents. The suggestion methodologies include LCS and Wu and Palmer in order to investigate alike for sentence pairs.

Keywords : Paraphrase Detection, plagiarism

Introduction

On the Merriam-Webster website, the term of word paraphrases is defined that “a restatement of a text, passage, or work giving the meaning in another form.” Therefore, paraphrase detection is basically used to determine the two or more sentences, which have similar content, or meaning. The syntax of paraphrases can be described into three forms, including synonyms, permutation, and topicality. To make more clear, three different types of paraphrase can be shown below (Per, 2010):

(1) Synonyms: $s_1 = w_1 + w_2 + \ldots w_n$ and $s_2 = w_1 + o_2 + \ldots w_n$

(2) Permutation $s_1 = w_1 + w_2 + \ldots w_n$ and $s_2 = w_2 + w_1 + \ldots w_n$

(3) Topicality $s_1 = w_1 + w_2 + \ldots w_n$ and $s_2 = x_1 + x_2 + \ldots x_n$

The notation for three sentences above can be described here: (1) the sentence S with j index; (2) the word W with i order; (3) the replacing word O; and (4) the unnecessary synonym word X.

Phrases below, which are taken from the Microsoft Research file (MSR file), represent the example of identical meaning from two sentences both English and Thai.

Sentences show in English:

S1. ["Senator Clinton should be ashamed of herself for playing politics with the important issue of homeland security funding," he said.]

S2. ["She should be ashamed of herself for playing politics with this important issue," said state budget division spokesman Andrew Rush.]

Sentences are translated to Thai:

S1. [“คลินตันวุฒิสมาชิกควรจะละอายใจของตัวเองสำหรับการเล่นการเมืองกับปัญหาที่สำคัญของเงินทุนมั่นคงแห่งมาตุภูมิ" เขากล่าว]

S2. [“เธอควรจะละอายใจของตัวเองสำหรับการเล่นการเมืองกับปัญหาที่สำคัญนี้" รัฐส่วนงบประมาณโฆษกแอนดรูรัชกล่าวว่า]

Related work

The paraphrase detection is the powerful weapon to fight with plagiarism, and it has been developing for a while. Cordeiro and colleague (Cordeiro et al., 2007) performed the Log-SimX function to figure out the limitations and outperforms for non-common cases and Web News Stories. They produced the final result by comparing the output from Asymmetrical Paraphrase function (AP function) with the output from Symmetrical Paraphrase function (SP function). (Long et al., 2006) developed two-phase process technique. Their methodology was used to find the identical words of sentence pair. After that, they compared this work to a simple lexical matching technique.

Another work is Semantic Text Similarity using Corpus-Based Word Similarity and String Similarity which was developed in (Islam et at., 2008). This work used Corpus-Based to measure similarity of texts. By combining three string similarity functions, which are LCS, MCLCS1 and MCLCSn, the new method was developed. To calculate the similarity score, Second Order Co-Occurrence PMI (SOC-PMI) was applied. Then, the calculation of complexity for each text was computed from the combination of two matrices.

Objectives

The objectives of this work consists of:

1. To establish detection tool for idea stealing in Thai academic works;
2. To develop fundamental resources for NLP using for Thai language.

Research Methodology

The operation of this research can be conducted as step below:

A. Preprocessing

Preprocessing is the first step to prepare data to work on document stealing the ideas in Thai documents. Preprocessing consists of six steps, and each step can be described below.

(1) MSR file will be translated to Thai, so-called Thai-MSR, and Google translator is the selected choice to make the job done. Translated sentences pairs in Thai-MSR will be used for the next step without correcting the statements from the translator. The translated sentences can be shown below.

S1 = “ประธานเจ้าหน้าที่ฝ่ายปฏิบัติการ PCCW ไมค์บุชเชอร์, และอเล็กซ์สังเวียนประธานเจ้าหน้าที่ฝ่ายการเงินจะรายงานตรงต่อฉันดังนั้น”

S2 = “ปัจจุบันประธานเจ้าหน้าที่ฝ่ายปฏิบัติการไมค์บุชเชอร์และหัวหน้าคณะผู้บริหารด้านการเงินอเล็กซ์อารีนาจะรายงานไปดังนั้น”

(2) Each full sentence is chopped to the word. Like step (1), converting sentence to work is working without correcting the result. An example can be seen below.

S1 = “|ประธาน|เจ้าหน้าที่|ฝ่าย|ปฏิบัติการ| |PCCW| |ไมค์|บุช|เชอร์|,| |และ|อเล็กซ์|สังเวียน|ประธาน|เจ้าหน้าที่|ฝ่าย|การเงิน|จะราย|งาน|ตรง|ต่อ|ฉัน|ดังนั้น|”

(3) Each sentence will eliminate the Thai stop words using Stop-word Removal (STR) technique.

(4) Both punctuation and number are obliterated.

(5) Sentence pairs are partitioned into array, S1 and S2.

(6) The duplicate words from pairs will be removed using tokenization technique.

B. Comparison the similarity between sentences

Three different subsequences are employed to measure the string similarity, including longest common subsequence (LCS), maximal consecutive longest common subsequence for the first start word (MCLCS1), and maximal consecutive longest common subsequence for starting at the on position (MCLCSn).

Algorithm 1. $MCLCS_1$ (Maximal Consecutive LCS starting at character 1)

input : r_i, s_j /*r_i and s_j are two input strings where $|r_i| = \tau$, $|s_j| = \eta$ and $\tau \leq \eta$ */

output: r_i /*r_i is the Maximal Consecutive LCS starting at character 1 */

1 $\tau \longleftarrow |r_i|, \eta \longleftarrow |s_j|$

2 **while** $|r_i| \geq 0$ **do**

3 **If** $r_i \cap s_j$ **then** /* i.e., $r_i \subset s_j = r_i$ */

4 **return** r_i

5 **else**

6 $r_i \longleftarrow r_i \backslash c_\tau$ /* i.e., remove the right most character from r_i */

7 **end**

8 **end**

Fig.1 $MCLCS_1$ (Maximal Consecutive LCS starting at character 1)

To describe MCLSC1 algorithm, two similar words, represented by W1 and W2, are played for the example. W1 contains “albastru” while W2 stores “alabaster”. When processing is working on MCLCS1 algorithm, three kinds of results can be produced and are shown below:

LCS(w1, w2) = ‘albastr’;

$MCLCS_1$(w1, w2) = ‘al’;

$MCLCS_n$(w1, w2) = ‘bast’.

The repeating of these steps will be utilized for the rest of words in a sentence (Fig. 2).

Algorithm 2. $MCLCS_n$ (Maximal consecutive LCS starting at any character *n*)

input : r_i, s_j /*r_i and s_j are two input strings where $|r_i| = \tau$, $|s_j| = \eta$ and $\tau \leq \eta$ */

output: *x* /*x is the Maximal Consecutive LCS starting at character *n* */

1 $\tau \longleftarrow |r_i|, \eta \longleftarrow |s_j|$

2 **while** $|r_i| \geq 0$ **do**

3 determine all *n-grams* from r_i where $n = 1 \ldots |r_i|$ and

4 $\overline{r_i}$ is the set of *n*-grams

5 **If** $x \in s_j$ where $\{x | x \in \overline{r_i}, x = Max(\overline{r_i})\}$ **then** /* *i* is the number of *n-grams* and $Max(\overline{r_i})$ returns the maximum length *n-gram* from $\overline{r_i}$ */

6 return *x*

7 **else**

8 $\overline{r_i} \longleftarrow \overline{r_i} \backslash x$ /*remove *x*| from r_i */

9 **end**

10 **end**

Fig.2 $MCLCS_n$ (Maximal consecutive LCS starting at any character n)

After the end of MCLCS processing, normalization, which can be seen from the several equations below, will be used for each subsequence following by NLCS (A), $NMCLCS_1$ (B), and $NMCLCS_n$ (C).

$$v_1 = NLCS(w1, w2) = \frac{length(LCS(w1,w2))^2}{length(w1) \times length(w2)} \text{----------(A)}$$

$$v_2 = NMCLCS(w1, w2) = \frac{length(MCLCS_1(w1,w2))^2}{length(w1) \times length(w2)} \text{----------(B)}$$

$$v_3 = NMCLCS(w1, w2) = \frac{length(MCLCS_n(w1,w2))^2}{length(w1) \times length(w2)} \text{----------(C)}$$

To illustrate, the computing of normalization for the NLCS, NMCLCS1, and NMCLCSn can be seen below:

NLCS(w1, w2) = $7^2/(8x9)$ = 0.68;

$NMCLCS_1(w1, w2) = 2^2/(8x9) = 0.056;$

$NMCLCS_n(w1, w2) = 4^2/(8x9) = 0.22.$

The results from above will be applied to the next step for weight calculation following mathematical definition of text similarity for two texts.

α = wg1v1 + wg2v2 + wg3v3 ----------(D)

The example of calculation of string similarity is shown:

α = wg1v1 + wg2v2 + wg3v3

= 0.33x0.68 + 0.33x0.056 + 0.33 x 0.22

= 0.32

After that, α_{ij} will be held into the matrix.

$$M_1 = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1(n-\delta)} \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2(n-\delta)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{i(n-\delta)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{(m-\delta)1} & a_{(m-\delta)2} & \cdots & a_{(m-\delta)j} & \cdots & a_{(m-\delta)(n-\delta)} \end{pmatrix}$$

C. Semantic Similarity between Words

To measure semantic similarity between words in this work, Wu and Palmer algorithm will be adapted to calculate β for textual similarity.

The description of Wu and Palmer method is indicated by:

$$\mathrm{sim}_{\mathrm{wup}} = \frac{2*\mathrm{depth(LCS)}}{\mathrm{depth(concept_1)+depth(concept_2)}} \text{----------(E)}$$

The result of β_{ij} will be come inside the matrix.

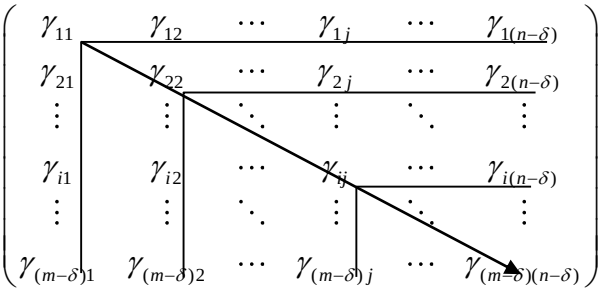
$$M_2 = \begin{pmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1j} & \cdots & \beta_{1(n-\delta)} \\ \beta_{21} & \beta_{22} & \cdots & \beta_{2j} & \cdots & \beta_{2(n-\delta)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \beta_{i1} & \beta_{i2} & \cdots & \beta_{ij} & \cdots & \beta_{i(n-\delta)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \beta_{(m-\delta)1} & \beta_{(m-\delta)2} & \cdots & \beta_{(m-\delta)j} & \cdots & \beta_{(m-\delta)(n-\delta)} \end{pmatrix}$$

D. Overall Sentence Similarity

The overall sentence similarity will be computed by creating n x m join matrix. The weight factor for both M1 and M2 matrix is 0.5, and γ is calculated inside the matrix.

$$M = \begin{pmatrix} \gamma_{11} & \gamma_{12} & \cdots & \gamma_{1j} & \cdots & \gamma_{1(n-\delta)} \\ \gamma_{21} & \gamma_{22} & \cdots & \gamma_{2j} & \cdots & \gamma_{2(n-\delta)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \gamma_{i1} & \gamma_{i2} & \cdots & \gamma_{ij} & \cdots & \gamma_{i(n-\delta)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \gamma_{(m-\delta)1} & \gamma_{(m-\delta)2} & \cdots & \gamma_{(m-\delta)j} & \cdots & \gamma_{(m-\delta)(n-\delta)} \end{pmatrix}$$

A result of the joint matrix is used to investigate a maximum value for each word pair.



All maximum values for each pair will be declared to ρ.

ρ = { Max_1, Max_2, … Max_n}

Finally, a total score will be produced by following this equation:

$$S(S1, S2) = \frac{\delta + \sum_{i=1}^{|\rho|} \rho_i \times (m+n)}{2mn} \text{----------(F)}$$

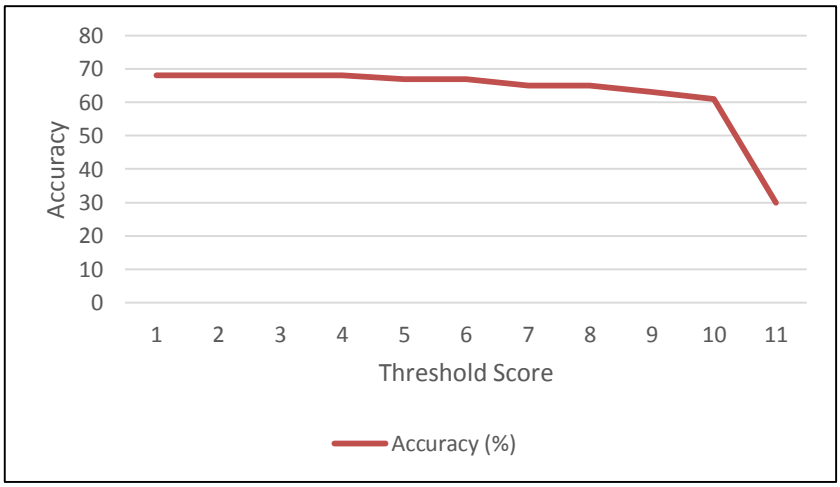
Results and Discussion

In order to test the system, the certain numbers of selecting sentence pairs from Thai-MSR were defined. The selecting sentences were included 100 and 500 sentence pairs out of 1,725 sentence pairs. The process for this system produced 11 results for each threshold score from 0, 0.1, 0.2, …, and 1.

For the first test, the selected 100 sentence pairs were used to test the method. The results were produced as shown in (Table I) and (Graph I).

Threshold Score	Accuracy (%)
0	68
0.1	68
0.2	68
0.3	**68**
0.4	67
0.5	67
0.6	65
0.7	65
0.8	63
0.9	61
1.0	30

Table I. The result for 100 sentence pairs.

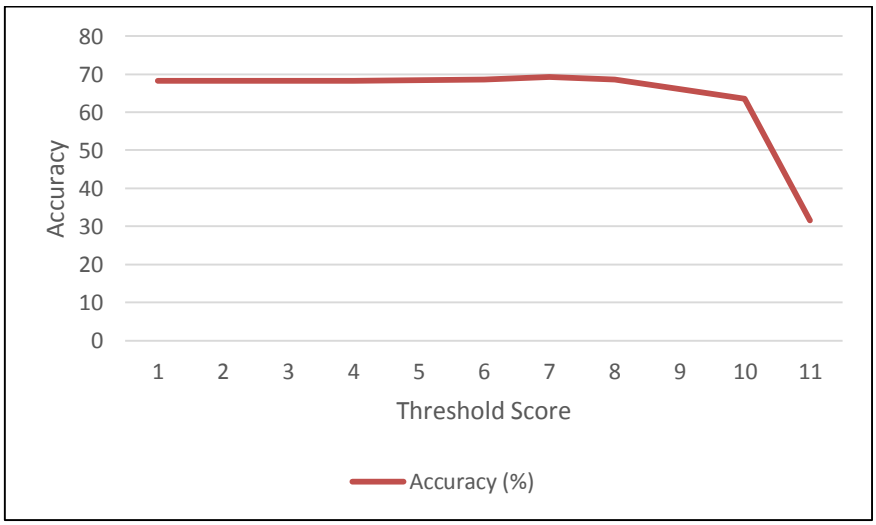


Graph I. The result for 100 text pairs.

The second selected 500 sentence pairs to test the method. The results were produced as shown in (Table II) and (Graph II).

Threshold Score	Accuracy (%)
0	68.2
0.1	68.2
0.2	68.2
0.3	68.2
0.4	68.4
0.5	68.6
0.6	**69.2**
0.7	68.6
0.8	66.0
0.9	63.6
1.0	31.6

Table II. The result for 500 sentence pairs.



Graph II. The result for 500 sentence pairs

A system produced a series of accuracy as 68% when a threshold score was set between 0 and 0.5 for 100 sentence pairs (Table I). By the contrast, testing for 500 sentence pairs given the best result, 69.2, when the threshold score was set to 0.6 (Table II). However, both two graphs shown a similar trending. In short, if threshold value was set between 6 and 8, a system

might have returned higher rate for accuracy. Some errors, nevertheless, were occurring because of no word left in pairs. In addition, if the calculation results were over than 1.00, sentence pairs were identical.

Conclusion

LCS family and Wu and Palmer can be used for detecting similarity for two Thai sentences. With designed 11 thresholds to test 100 and 500 sentence pairs, a system can give the best result if a threshold value is not set more than 0.8. The worst results, by contrast, could come out if a threshold value is set to 1.

For the future work, a system should apply ontology technique, which is popularly used for textual similarity. In addition, the larger data set and experimentation for multiple times will be applied.

References

Cordeiro J., Dias G. and Brazdil P. (2007). New Functions for Unsupervised Asymmetrical Paraphrase Detection, 2(4):12-23.

Islam, A. and Inkpen, D. (2008). Semantic text Similarity Using Corpus-Based Word Similarity and String Similarity. 2(2)

Long Q. Min-Yen K. and Tat-Seng C. (2006). Paraphrase recognition via dissimilarity significance classification: 18-26

Per A. (2010). Plagiarism Detection: Experiments to investigate the utility of linguistically Informed Features in detection Textual Plagiarism. Swedish Institute of Computer Science.