IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 10, No. 3, September 2021, pp. 636~648
ISSN: 2252-8938, DOI: 10.11591/ijai.v10.i3.pp636-648

A two-phase plagiarism detection system based on multi-layer long short-term memory networks

Nguyen Van Son[1], Le Thanh Huong[2], Nguyen Chi Thanh[3]
[1,3]Institute of Information Technology, MIST, Vietnam
[2]School of Information and Computer Science Technology, Hanoi University of Science and Technology Hanoi, Vietnam

Article Info

Article history:

Received Aug 25, 2020
Revised May 15, 2021
Accepted May 26, 2021

Keywords:

Deep learning
Feature extraction
Multi-layer long short-term memory
Plagiarism detection
Two-phase

ABSTRACT

Finding plagiarism strings between two given documents are the main task of the plagiarism detection problem. Traditional approaches based on string matching are not very useful in cases of similar semantic plagiarism. Deep learning approaches solve this problem by measuring the semantic similarity between pairs of sentences. However, these approaches still face the following challenging points. First, it is impossible to solve cases where only part of a sentence belongs to a plagiarism passage. Second, measuring the sentential similarity without considering the context of surrounding sentences leads to decreasing in accuracy. To solve the above problems, this paper proposes a two-phase plagiarism detection system based on multi-layer long short-term memory network model and feature extraction technique: (i) a passage-phase to recognize plagiarism passages, and (ii) a word-phase to determine the exact plagiarism strings. Our experiment results on PAN 2014 corpus reached 94.26% F-measure, higher than existing research in this field.

Corresponding Author:

Nguyen Van Son
Institute of Information Technology
MIST, Vietnam
Tel: (+84) 904236683
Email: sonnv78@gmail.com

1. INTRODUCTION

Plagiarism is defined as the reuse of another person's ideas, processes, results, or words without explicitly acknowledging the source [1]. Plagiarism detection is the algorithm for automatically retrieving strings in a suspicious document reused from another document. Plagiarism methods are divided into two main types: literal plagiarism and intelligent one, based on the plagiarist's behavior [2]. Literal plagiarism is a common and popular case in which plagiarists do not spend much time hiding the academic crime they committed. For example, they copy and paste the text from the internet. Intelligent plagiarism is severe academic dishonesty wherein plagiarists try to deceive readers by changing others' contributions to appear as their own. Intelligent plagiarists try to hide, obfuscate, and change the original work in various intelligent ways, including text manipulation, translation, and idea adoption.

Over the past two decades, automatic plagiarism detection has received significant attention from the research community. Two main tasks of automatic plagiarism detection are source retrieval and text alignment. In the source retrieval task, given a suspicious document and a web search engine, the task is to retrieve all source documents from which text has been reused. In the text alignment subtask, given a pair of documents (a suspicious document and a source one), the task is to identify contiguous maximal-length passages of reused text.

Most of existing works on text alignment focus on supervised and unsupervised approaches. Several unsupervised approaches use character-based methods (e.g., [1], [3], [4]) that applied string matching or approximate string matching with measures such as Hamming or Levenshtein distances to compute the similarity between two strings within a sliding window. Instead of comparing strings as in character-based methods, vector-based methods (e.g., [5], [6]) proposed representing input texts as vectors of tokens and measuring the distance between these vectors by using similarity coefficients such as Jaccard, Cosine, Euclidean, or Manhattan distances.

Based on the intuition that similar documents would have similar syntactical structures, some research works (e.g., [7], [8]) used syntactic information at the first stage of measuring sentential similarity. The main limitation of these unsupervised approaches is that they cannot deal with intelligent plagiarism in which the same content can be expressed by different words and in different orders. Research on intelligent plagiarism (e.g., [9]-[11]) often concentrate on finding the similarity between pairs of sentences. Gharavi *et al.* [9] proposed a plagiarism detection method for the Persian language by representing each sentence by a semantic embedding vector and then comparing the similarity between these vectors using the cosine similarity.

Cherroun *et al.* [10] proposed a two-phase system using a supervised learning approach to detect plagiarism in Arabic. The first phase produced a representing vector for each sentence by combining different features, including word embedding, word alignment, term frequency weighting, and part-of-speech tagging. The second phase used lexical, syntactic, and semantic features in three machine learning models (support vector machine (SVM), decision trees (DT), and random forests (RF)) to improve the accuracy of the first phase results. However, their approach did not deal with obfuscated plagiarism cases when a passage is inserted in the middle of a sentence. Altheneyan *et al.* [11] presented two systems (PlagLinSVM and PlagRbfSVM) using the support vector machine classifier (SVM) with lexical, syntactic, and semantic features to detect plagiarism sentences. Their approach applied two plagiarism detecting levels: paragraph and sentence ones. The paragraph-level detects similar paragraphs in the two input documents basing on the number of common unigrams and bigrams of these paragraphs. The sentence-level aligns sentences in the above result paragraph pairs basing on the number of common unigrams between the two sentences. If the score of a sentence pair was higher than the pre-defined threshold, the SVM classifier is applied to determine whether two sentences are similar or not. Finally, plagiarism passages were created by connecting adjacent sentences that were copied from the source documents.

Previous intelligent plagiarism approaches have limitations on finding copied paragraphs based on sentence units, assuming that people only copy or rewrite sentences. However, existing cases of plagiarism are more complicated than that. When comparing the plagiarism strings and the source one, we found that they can be different in; (i) the number of sentences; (ii) the sentence length; and (iii) the text appearance's order. The above situations are not resolved yet in existing research on plagiarism detection.

Recently, deep learning approaches have proven to be efficient in solving many tasks of natural language processing. However, as far as we know, the largest training corpus for the plagiarism detection task is still very small for the training phase. Therefore, in this paper, we propose a plagiarism system that takes advantage of hand-crafted feature vectors and long short-term memory (LSTM) network model [12] to deal with the problems mentioned above. The system includes two main phases:

- passage-phase to figure out plagiarism passages in suspicious and source documents.
- word-phase to remove redundancy parts from plagiarism passages to achieve the exact plagiarism strings.

The main contributions of this work are:

- We proposed new features at both the passage and word level to improve the accuracy in detecting similar strings between two documents. These features are: (i) Maximize passage similarity, maximize passage intersection, passage importance at the passage-phase; and (ii) word similarity, average word similarity, sentence based similarity at the word-phase.
- We proposed a two-phase plagiarism detection system based on a multi-layer LSTM network model using our proposed features to solve both literal and intelligent plagiarism problems.

The rest of the article is organized as: our proposed method is introduced in section 2. In section 3, we describe our experiments and analyze the results. Finally, our conclusions and future research directions are presented in section 4.

2. PROPOSED METHOD

The problem of finding similar strings between two documents is stated is [13]:

Definition 1: Given two documents d and d', the goal is to detect a set of passage pairs, P, such as:

$$P = \left\{ < p_{d_i}, p_{d'_j} > | \forall\, p_{d_i}, \forall p_{d'_j} : p_{d_i} \in d \,\Lambda\, p_{d'_j} \in d' \Lambda \left| p_{d_i} \cap p_{d'_j} \right| > \delta \right\} \quad (1)$$

in which p_{d_i} is a string from d; $p_{d'_j}$ is a string from d'; $p_{d_i} \cap p_{d'_j}$ indicates the similarity between p_{d_i} and $p_{d'_j}$; δ is a threshold that is used to determine whether two strings are similar enough to be considered as plagiarism.

The series of competition shared tasks for plagiarism detection named plagiarism analysis, authorship identification, and near-duplicate detection (PAN) has defined four types of plagiarism.

a. None obfuscation: Create plagiarism cases by copying a paragraph from the source document and insert it into the suspicious one.
b. Random obfuscation: Create plagiarism cases by inserting, deleting, changing the order of words from a paragraph of the source, and inserting it into the suspicious document.
c. Translation obfuscation: Create plagiarism cases by translating a paragraph more than once through several languages and back to the original language using different machine translation tools. Then, inserting the translated paragraph into the suspicious document.
d. Summary obfuscation: Create plagiarism cases by summarizing the source paragraph and inserting it into the suspicious document.

This paper aims at solving plagiarism cases belong to all four types above. Our proposed system's workflow is shown in Figure 1, including three steps.

- Pre-processing: This step splits input documents into sentences, removes stopwords and special characters, and combines sort sentences into one.
- Passage-phase: After the pre-processing step, we use a context window sliding over the source and suspicious documents to create candidate passages. We extract features from these passages and generate an input feature matrix corresponding to these features. This matrix is feed into a binary classifier of the candidate selection module to obtain pairs of plagiarism passages.
- Word-phase: The pairs of plagiarism passages are used as the input for the word-phase. The purpose of this phase is to define the exact plagiarism strings from the input passages. A binary classifier at the word-level is used to perform this task.

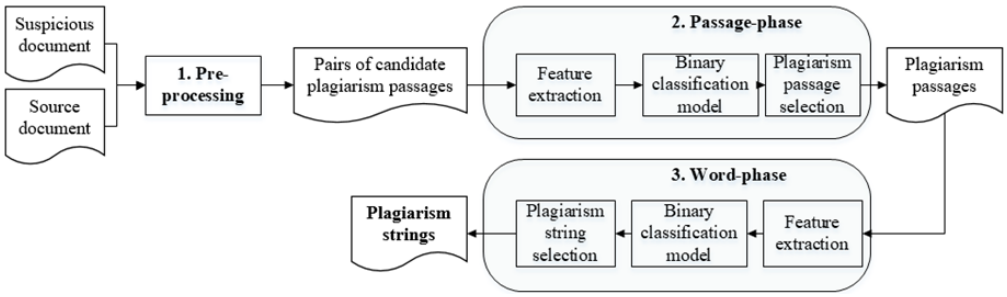

Figure 1. Overview of the proposed system's workflow for plagiarism detection

2.1. Pre-processing

The input documents are split into sentences using the sent tokenizer tool from the NLTK library. Then stopwords are removed from these sentences. Some specific cases can affect the accuracy of plagiarism selection. These cases are:

- The input documents contain numbers that are written incorrectly, such as '8. 39', '7 p. m'. In this case, the sentence splitter incorrectly segments text into sentences at the dot ('.') character.
- After removing stopwords, there are some short sentences containing none or only one or two tokens. For example, two sentences "Can you feel the burn?", "Who we are?" remain two words and empty, respectively, after cleaning stopwords and punctuation characters.

Since the similarities of short sentences do not have much meaning, we combine the short sentences with surrounding sentences and compare the similarity between the passages after combined. Therefore, to deal with the problems mentioned above, we first apply the sentence splitter and then remove stopwords, numbers, and special characters from the sentences. After cleaning the text, sentences with less than three words are combined with the next sentence to create extended sentences. To the best of our knowledge, the above combination step allows us to efficiently manage the passage's length after pairing and avoiding the case of creating too-long passages. We use a window of size w (sentences) sliding on both suspicious and

source documents to generate candidate plagiarism passages, which are used as inputs of the passage-phase. The optimal window size for the PAN datasets is three sentences.

2.2. Passage-phase

The input of this phase is candidate plagiarism passages, each passage consisting of three consecutive sentences from the suspicious or source documents. In this phase, each passage is encoded as a semantic embedding vector. The semantic similarity between two passages is calculated based on the distance between these vectors. We use SBERT to encode passages, since it is proved in [14] that SBERT is better than other methods (e.g., Word2Vec [15], Glove [16], Fastext [17], InferSent [18], or Universal Sentence Encoder [19]) in various domains. Features representing for each passage is derived from these passage vectors. They are then used as inputs for the binary classification at the passage level to detect whether two passages are similar or not.

2.2.1. Passage-phase feature extraction

Given a set of all candidate passages in the suspicious document $U = (u_1,u_2,\ldots,u_n)$ and a set of all candidate passages in the source document $V = (v_1,v_2,\ldots,v_m)$, with each passage u_i and v_j is represented as a passage embedding vector. We propose the following features for this phase:

– Maximize passage similarity

This feature is used to determine the maximum similarity of a passage vector u_i against a set of passage vectors *V*. Let us say sim_{u_i,v_j} is the similarity between two passage vectors u_i and v_j where $u_i \in U$, $v_j \in V$. Let $sim_{u_i,V}$ is the maximum passage similarity of the passage vector u_i against the set of passage vectors *V*. It is calculated as:

$$sim_{u_i,V} = \max_{v_j \in V} cosin(u_i, v_j) \quad (2)$$

The *maximize passage similarity* feature vector of all passage vectors in the pair of suspicious and source document is determined by (3):

$$psim(U,V) = (sim_{u_1,V}, sim_{u_2,V}, \ldots, sim_{u_n,V}, sim_{v_1,U}, sim_{v_2,U}, \ldots, sim_{v_m,U}) \quad (3)$$

– Maximize passage intersection

To determine the maximum intersection value of a passage u_i with a set of passages *V*, we split passages into words and find the intersection words of each passage pair (u_i, v_j), with $u_i \in U$, $v_j \in V$ and take the maximum length of this intersection. This value is calculated as in (4):

$$inter_{u_i,V} = \max_{v_j \in V} len(u_i \cap v_j) \quad (4)$$

The *maximize passage intersection* feature vector of all passages in the pair of suspicious and source document is determined by (5):

$$pinter(U,V) = (inter_{u_1,V}, inter_{u_2,V}, \ldots, inter_{u_n,V}, inter_{v_1,U}, inter_{v_2,U}, \ldots, inter_{v_m,U}) \quad (5)$$

– Passage importance

Term frequency-inverse document frequency (TF-IDF) is the most widely used and considered one of the most appropriate term weighting schemes. This TF-IDF is employed to get rid of terms with lower weights from documents and helps to increase the retrieval effectiveness. Term frequency-inverse document frequency is a numerical statistic that tells us how important a word is to a document in a collection or a corpus. It is mostly used as a weighting factor in various processes used for information retrieval and text mining. To determine similar passages, we put forward the idea of term frequency-inverse sentence frequency (TF-ISF) [20]. We treat each passage as a document and each document as a corpus, then calculate the values of *TF(w,U)*, *TF(u_i,U)*, and *ISF(u_i,U)*, in which *w* is a term in a passage u_i, *U* is the document containing u_i. Given $|u_i|$ is the total number of words in the passage u_i, *TF(u_i,U)* is computed as:

$$TF(u_i, U) = \frac{\sum_{w \in u_i} TF(w,U)}{|u_i|} \quad (6)$$

ISF(u_i,U) is computed by (7):

$$ISF(u_i, U) = \frac{\sum_{w \in u_i} IDF(w,U)}{|u_i|} \quad (7)$$

The passage importance of the passage u_i in the document U is determined by (8):

$$imp_{u_i,U} = TF(u_i, U) \times ISF(u_i, U) \quad (8)$$

The *passage importance* feature vector of all passage in the pair of suspicious and source document is determined by (9):

$$pimp(U,V) = (imp_{u_1,U}, imp_{u_2,U}, \dots, imp_{u_n,U}, imp_{v_1,V}, imp_{v_2,V}, \dots, imp_{v_m,V}) \quad (9)$$

– The feature matrix for the passage-phase

After extracting and creating three feature vectors *psim(U,V)*, *pinter(U,V)*, and *pimp(U,V)*, we combine them into a two-dimensional matrix of size (n+m) x 3 where n+m is the total number of passages from suspicious and source documents. The feature matrix for all passages in the pair of suspicious and source documents is determined as in (10). It is used as the input for the multi-layer LSTM network model, described in section 2.2.2.

$$f_{passage} = \begin{pmatrix} sim_{u_1,V} & inter_{u_1,V} & imp_{u_1,U} \\ sim_{u_2,V} & inter_{u_2,V} & imp_{u_2,U} \\ \vdots & \vdots & \vdots \\ sim_{v_m,U} & inter_{v_m,U} & imp_{v_m,V} \end{pmatrix} \quad (10)$$

2.2.2. Plagiarism passage selection

We build our binary classifier by using a multi-layer LSTM network model, which is used to predict the probability of being a plagiarism passage in the pair of suspicious and source documents. Figure 2 shows the structure of our model at the passage-phase. At this phase, we generate the input vectors by reshaping the feature matrix $f_{passage}$ into a three-dimensional matrix of *batch_size*, *time_steps*, and *seq_len* and feed them into the model. The parameters using in the LSTM model are: (i) *batch_size* equals the number of passages; (ii) *time_steps* equals 1; (iii) *seq_len* equals the number of features (seq_len=3).

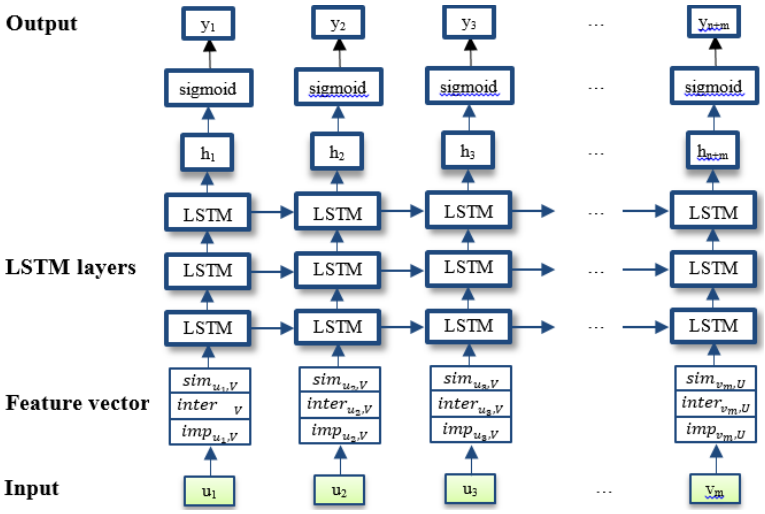

Figure 2. The architecture of the multi-layer LSTM model at the passage-phase

The output of the sigmoid activation function is always in the range of (0,1). This function is applied to the output of all units in the last hidden LSTM layer. Let $y = (y_1, y_2, \dots, y_{n+m})$ is the output of the binary classification model ($0 < y_i < 1$), and n+m is the number of passages in the pair of suspicious and source documents. Figure 3 shows the output of the model is a vector of 0s and 1s in which values 1 for all y_i being higher than a threshold θ, and values 0 for the remaining.

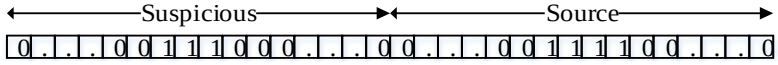

Figure 3. The output of the model at the passage-phase

Plagiarism passages are generated by selecting sentences corresponding to the longest values of 1 from the output of the model. When observing and analyzing the plagiarism passages obtained, we found that most plagiarism passages contain entire sentences. However, the plagiarism paragraph contains several redundant words at the two ends, such as the example in the PAN 2014 corpus explained by: this example. In this example, the underlined text is inside the plagiarism paragraph, whereas the rest is redundant.
The suspicious plagiarism paragraph:

> The capsule was designed for entry into the Martian atmosphere, descent to the surface, impact survival, and surface lifetimes of as much as six months and contained the power, guidance, control communications, and data handling systems necessary to complete its mission. *is perhaps the most productive space probe yet deployed, visiting four planets and their moons, including two primary visits to previously unexplored planets, with powerful cameras and a multitude of scientific instruments, at a fraction of the money later spent on specialized probes such as the and the probe. Along with, and Voyager 2 is an .Voyager 2 Galileo spacecraft Cassini-Huygens [2] [3] Pioneer 10 Pioneer 11 Voyager 1 New Horizons interstellar probe* resident per year, or roughly half the cost of one candy bar each year since project inception.

The source plagiarism paragraph:

> Voyager 2 unmanned interplanetary space probe Voyager program Voyager 1 Voyager 2 ecliptic Solar System Uranus Neptune gravity assist Saturn Voyager 2 Titan Planetary Grand Tour [1] *is perhaps the most productive space probe yet deployed, visiting four planets and their moons, including two primary visits to previously unexplored planets, with powerful cameras and a multitude of scientific instruments, at a fraction of the money later spent on specialized probes such as the and the probe. Along with, , and Voyager 2 is an .Voyager 2 Galileo spacecraft Cassini-Huygens [2] [3] Pioneer 10 Pioneer 11 Voyager 1 New Horizons interstellar probe* Contents Titan 3E Centaur was originally planned to be, part of the.

To solve this problem, we extend pairs of plagiarism passages from the suspicious and source documents by adding *k* sentences to the left and right of both passages. Extended passages will be used as the input for the word-phase to find exact plagiarism strings. It is done by removing redundant text from the extended plagiarism passages. The word-phase will be introduced next.

2.3. Word-phase

To remove the redundant text at the two ends of the extended plagiarism passages, we need to identify semantically related segments based on consecutive words of high similarity. To get the meaning of a word, we put that word in a window size of 3 with one word on the left and one word on the right. The text inside this window is used as the input of SBERT to create word feature vectors.

2.3.1. Word-level feature extraction

In this phase, three features are proposed based on the cosine similarity between the word and the sentence containing that word. The *word similarity* feature is a vector that contains the maximum similarity values of each word. The maximum similarity of a word in the suspicious passage is the maximum similarity of that word with each word in the source passage and vice versa. Features *average word similarity* and *sentence based similarity* are used to solve cases where the similarity value of a word has a big difference with the surrounding words. The *average word similarity* feature is a vector that each item is the average of the *word similarity* values within the sentence. The *sentence based similarity* feature is a vector that each item is the maximum of sentence similarities of the sentence containing that word. The detailed information on the word-phase features is explained by:

Given the extended suspicious passage $P=(p_1,p_2,\ldots,p_n)$, the extended source passage $Q=(q_1,q_2,\ldots,q_m)$ with each word p_i and q_j is represented by a word embedding vector.

– Word similarity

Let us call $sim(p_i,q_j)$ is the cosine similarity between two word vectors p_i and q_j. The *word similarity* feature between P and Q is a vector being computed as (11).

$$wsim(P,Q) = (\max_{q_j \in Q} sim(p_1, q_j), \max_{q_j \in Q} sim(p_2, q_j), \ldots, \max_{p_j \in P} sim(q_m, p_j)) \quad (11)$$

– Average word similarity

Given wi (with i= 1÷n+m), is the i-th word in the pair of suspicious and source passages, d is the sentence that $w_i \in d$, and $|d|$ is the total number of words in the sentence *d*. Let us call $avg(w_i)$ is the *average similarity* of word w_i in the sentence *d; wsim(i)* is the value of the *i*-th item in the *word similarity* feature vector. Then, the $avg(w_i)$ is computed as:

$$avg(w_i) = \frac{\sum_{w_i \in d} wsim(i)}{|d|} \quad (12)$$

The *average word similarity* feature between two passages P and Q is a vector determined by the following formula:

$$wavg(P,Q) = (avg(p_1), avg(p_2), \ldots, avg(p_n), avg(q_1), avg(q_2), \ldots, avg(q_m)) \quad (13)$$

– Sentence based similarity

We reuse the *maximize passage similarity* feature (as described in the passage-phase) with the meaning of the passage is the sentence. Given the set of sentences U = (u_1,u_2,…,u_k), and V = (v_1,v_2,…,v_s) in the suspicious and source passages, respectively. Let us call $sim\_sent(p_i)$ is the *sentence based similarity* of word p_i in the sentence u_j. The $sim\_sent(p_i)$ is computed as:

$$sim\_sent(p_i) = \max_{v_l \in V} cosin(u_j, v_l) \mid \forall p_i \in u_j \quad (14)$$

The *sentence based similarity* feature between two passages P and Q is a vector determined by the following formula:

$$wsent(P,Q) = (sim\_sent(p_1), sim\_sent(p_2), \ldots, sim\_sent(p_n), sim\_sent(q_1), sim\_sent(q_2), \ldots, sim\_sent(q_m)) \quad (15)$$

The feature matrix for the word-phase:

After computing three feature vectors *wsim(P,Q)*, *wavg(P,Q)*, and *wsent(P,Q)*, we combine these feature vectors into a two-dimensional matrix of size *(n+m) x 3*.

$$f_{word} = \begin{pmatrix} \max_{q_j \in Q} \mathrm{sim}(p_1, q_j) & avg(p_1) & sim\_sent(p_1) \\ \max_{q_j \in Q} \mathrm{sim}(p_2, q_j) & avg(p_2) & sim\_sent(p_2) \\ \vdots & \vdots & \vdots \\ \max_{p_j \in Q} \mathrm{sim}(q_m, p_j) & avg(q_m) & sim\_sent(q_m) \end{pmatrix} \quad (16)$$

The feature matrix of all the extended plagiarism passages is determined by (16). This feature matrix is used as the input for the multi-layer LSTM model, described in section 2.3.2.

2.3.2. Plagiarism string selection

In this section, we conduct two processing steps: (i) *select plagiarism sentences* and (ii) *remove redundant text*. The details of each step are described as:

– Select plagiarism sentences

To select exact plagiarism sentences from the extended plagiarism passages, we use a multi-layer LSTM model whose input is taken from the feature matrix f_{word} as shown in Figure 4. The parameters using in this model are: (i) *batch_size* equals the number of words; (ii) *time_steps* equals 1; (iii) *seq_len* equals the number of features (seq_len=3).

In Figure 4, p_i and q_j denotes the *i*-th and *j*-th word in the pair of extended plagiarism passages, $y\_pred = (y_1, y_2, \ldots, y_{n+m})$ is the output of the binary classification model *($0 < y_i < 1$), n+m* is the total number of words in the pair of these passages. The predicted mean value of a sentence *u* is computed as in (17):

$$y\_pred\_sent_u = avg(y\_pred_u) = \frac{\sum_{w_i \in u} y_i}{|u|} \quad (17)$$

where w_i is a word in the sentence u.

After computing values $y\_pred\_sent_u$ for all sentences, we create a vector with the size corresponding to the total number of sentences in the pair of plagiarism passages. If the value of *y_pred_sent* of a sentence is higher than a threshold β, the value corresponding to that word in the sentence is 1; otherwise, it is 0. We select the longest strings with the value of 1 as the *plagiarism sentences.*

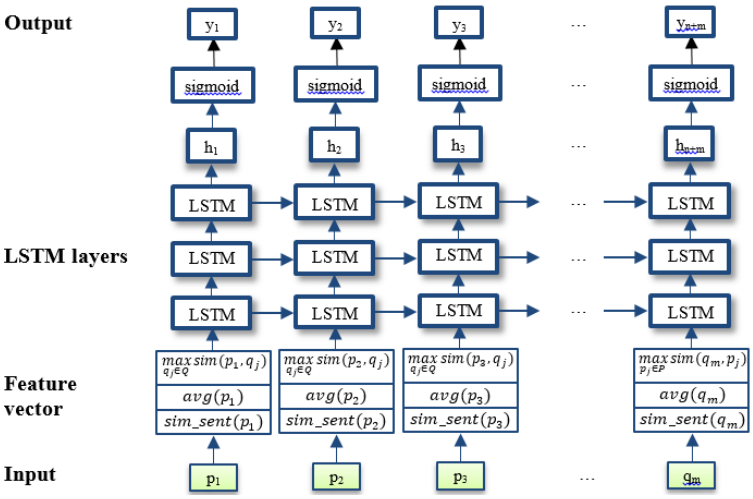

Figure 4. The architecture of the multi-layer LSTM model at the word-phase

– Remove redundant text

To achieve the exact plagiarism strings, we consider the leftmost plagiarism sentence and the rightmost one. The difference between these sentences' *max_threshold* and *min_threshold* is higher than t_1 (t_1=0.4). The *max_threshold* and *min_threshold* of a sentence u are determined by (18) and (19):

$$max\_threshold_u = \max_{w_i \in u} y_i \quad (18)$$

$$min\_threshold_u = \min_{w_i \in u} y_i \quad (19)$$

with w_i is a word in the sentence u.

These sentences above have one part inside and the remaining part outside the plagiarism passage. The outside part is on the left *(orient =1*) if the sentence is on the left of the plagiarism sentences or on the right *(orient =2)* if the sentence is on the right of the plagiarism sentences. If the previous step result contains only one sentence, the outside part belongs to the two ends *(orient =3)* of the sentence. Analyzing the output vector of the LSTM model *y_pred*, we discover that the predicted value y_i corresponding of the inside words is much higher than the predicted value y_j corresponding of the outside ones.

Algorithm 1 is used to cut off the redundant text from these sentences. The idea of this algorithm is: Given a threshold α, find the longest text in the leftmost sentence and the rightmost one whose all of their words have the predictive value *y_pred* < α. We defined the left and right position as the first and last word of the exact plagiarism strings, respectively. The algorithm receives the following parameters as inputs:

– *y_d*: is the predicted vector of the sentence. $y\_d = (y\_d_1, y\_d_2, \dots, y\_d_t)$ with *t* is the number of words in the sentence.
– *orient:* determines the intersection position in the left (*orient =1*) or right (*orient =2*) or both sides (*orient =3*) of boundary sentences.

Algorithm 1: Intersection position determination

```
Input: y_d, orient
    # orient = 1: left; orient = 2: right; orient = 3: both
    pos_left = 0; pos_right = length(y_d) - 1
    α = min(y_d) + (max(y_d)-min(y_d))/2
    if orient = 1 or orient = 3 then
        for i = 0 to length(y_d) - 1 do
            if y_d [i] > α then
```

```
7:                    pos_left = i
8                     break
9:    if orient = 2 or orient = 3 then
10:         for i = length(y_d) – 1 downto 0 do
11:             if y_d[i] > α then
12:                 pos_right = i
13                  break
Output: pos_left, pos_right
```

We initialize the left and right positions with the first and last points, respectively (lines 2). The threshold α is the average value of maximum and minimum of *y_d* vector. We define the left (line 4) and right (line 9) position based on the *orient* value. For each direction, we scan all the points (line 5 and line 10) and get the first points whose predict value *y_pred* are higher than the threshold α (line 7 and line 13). These points are the results of the algorithm.

3. EXPERIMENT RESULTS AND DISCUSSION

In our experiment, we use PAN 2013 text alignment training corpus [21] for training the system. This corpus is also the training corpus using in PAN 2014 competition. The PAN 2013 corpus consists of 1000 no obfuscation, 1000 random obfuscation, 1000 translation obfuscation, and 1185 summary obfuscation pairs of documents. Normally, this corpus is too small for training a deep learning model. By our experiment, we will prove that our approach of combining hand-crafted features with the LSTM model will be a good solution for this problem. To compare our system performance with state-of-the-art research in this task, we used PAN 2014 text alignment test corpus [22] for evaluating the system.

3.1. Evaluation metrics

Our system was evaluated by using a tool provided by PAN to measure the system performance. Four measures used in PAN are macro-averaged Precision, Recall, Plagdet, and Granularity. The formula to compute these values are described such as:

Given *S, R, s, r* are a set of all plagiarism cases, a set of all plagiarism system-detection cases, a plagiarism case, and a plagiarism system-detection case, respectively. The macro-averaged precision and recall are defined by:

$$prec(S,R) = \frac{1}{|R|} \times \sum_{r \in R} \frac{|\cup_{s \in S}(s \cap r)|}{|r|} \quad (20)$$

$$rec(S,R) = \frac{1}{|S|} \times \sum_{s \in S} \frac{|\cup_{r \in R}(s \cap r)|}{|s|} \quad (21)$$

The detection granularity of R under S indicates whether each plagiarism case s ∈ S is detected as a whole or in several pieces. It is calculated as:

$$gran(S,R) = \frac{1}{|S_R|} \times \sum_{s \in S_R} |R_S| \quad (22)$$

where $S_R \subseteq S$ are cases detected by detections in R, and $R_S \subseteq R$ are the detections of a given s.

Plagdet is the overall score of the system, which is calculated as:

$$plagdet(S,R) = \frac{2 \times prec \times rec}{prec + rec} \times \frac{1}{log_2(1 + gran(S,R))} \quad (23)$$

3.2. Experimental results and analysis

Several tests have been carried out to choose the best configuration for our system. We performed experiments by each phase to optimize parameters of the system. Extracted feature vectors from pairs of documents in the PAN 2013 training corpus are passed to the multi-layer LSTM model during the training process. We chose *binary_crossentropy* as the loss function since the model is a binary classification model. The threshold θ, which is used to select sentences in the passage-phase, is chosen to be *0.1*. To choose the value k (mentioned in section 2.2.2) for extending plagiarism passages, we initiate the k value by 1 and continuously increasing this value until the system reaches the highest recall value. Experiments proved that the value of *k* depends on the length of the plagiarism passages, as shown in Table 1.

At the word-phase, instead of using thresholds to identify each word, we apply the threshold β (β = *0.1)* to the *y_pred_sent*. The LSTM model generates an array whose size is equal to the number of

sentences. The value of the array's element is 1 if *y_pred_sent* is higher than *β*, and 0 for others. Then we select a continuous string with the highest predicted value. Table 2 shows the accuracy and loss values in the LSTM training phase with the four datasets in PAN 2013.

To evaluate the effectiveness of our proposed features, we carried experiments using each feature instead of all features, with the input is pairs of documents from PAN 2014 test corpus. Figure 5 shows the effect of these features at the word-phase on the system output. Three pairs of Figures 5(a) to 5(f) show the prediction results of *y_pred* and the final results using 1, 2, and 3 features, respectively. In these figures, the blue line shows the predicted result; the red line shows the average predicted value by sentences. The green line separates the suspicious and source passage; the black line shows the range of the selected plagiarism passages. The evaluation results proved that all the proposed features are useful, solving well for both literal plagiarism and intelligent plagiarism.

Table 1. The dynamic parameters for extending passage

	Plagiarism passage's length	k
1	≥6 sentences	1
2	≥3 sentences	2
3	≥2 sentences	3
4	1 sentence	4

Table 2. Accuracy and loss values of the training phase

PAN 2013 training corpus	Sentence level		Word level	
	Accuracy	Loss	Accuracy	Loss
None Obfuscation	0.9925	0.0068	0.9808	0.0161
Random Obfuscation	0.9727	0.0814	0.9303	0.1909
Translate Obfuscation	0.9707	0.0748	0.9443	0.1229
Summary Obfuscation	-	-	0.9201	0.2096

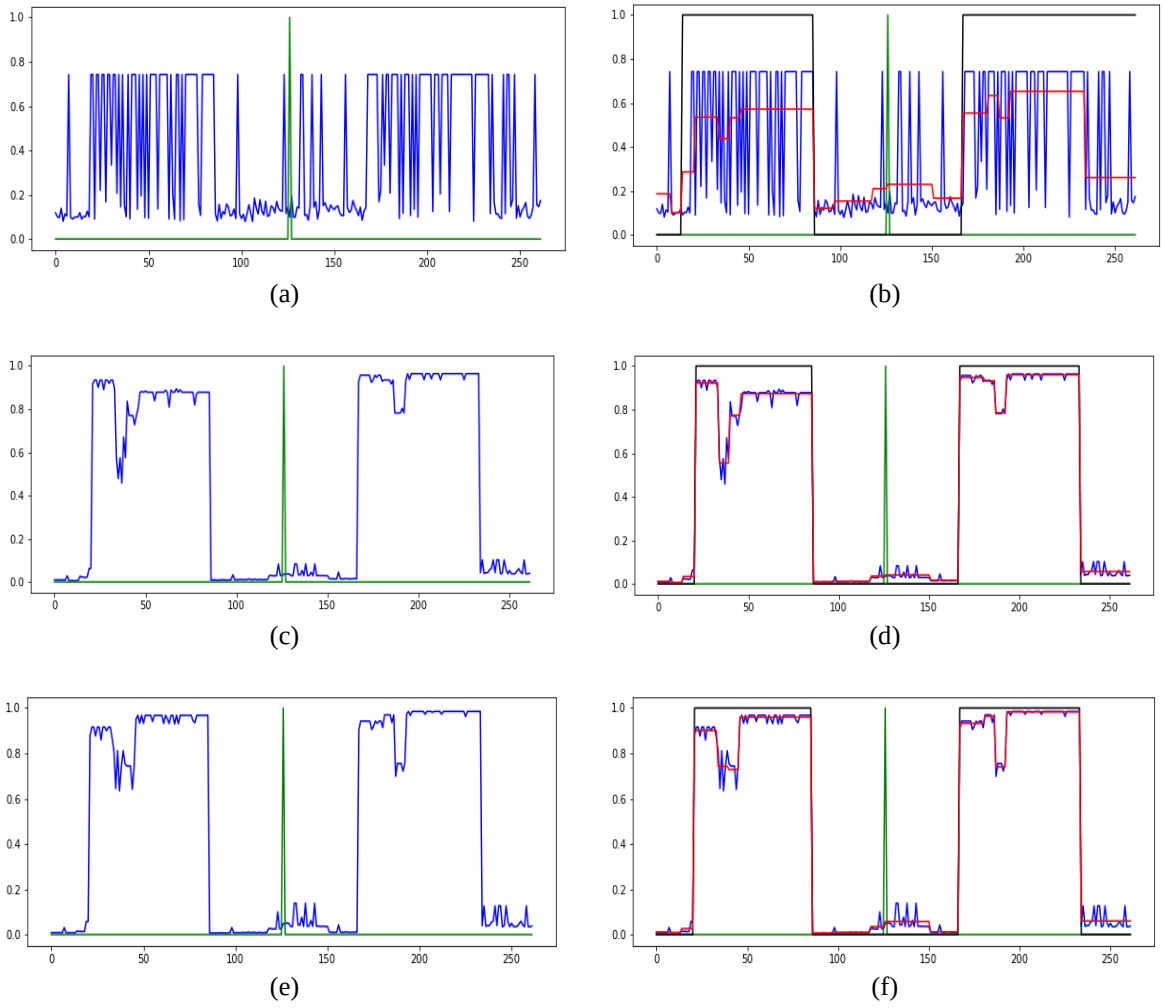

Figure 5. Effects of selecting different features at word-phase to plagiarism passage: (a) using one feature-wsim (P,Q); (b) output's result when using wsim (P,Q); (c) using two features-wsim (P,Q), wavg (P,Q); (d) output's result when using wsim (P,Q), wavg (P,Q); (e) using three features-wsim (P,Q), wavg (P,Q); wsent (P,Q); and (f) output's result when using wsim (P,Q), wavg (P,Q), wsent (P,Q)

Table 3 compares our system performance compared with existing research on this task using PAN 2014 as the test set. It shows that our system has a remarkable improvement comparing to other researches. It indicates that our system can detect most plagiarism cases comparing to others. The results prove that our proposed feature extraction techniques combining with our LSTM models is a promise solution for the case of detecting intelligent plagiarism in which the same content can be expressed in different ways and by different words, using a small training corpus.

Table 3. Performance comparison with state-of-the-art approaches

Team	F-measure (%)	Plagdet (%)	Prec (%)	Rec (%)	Gran
Our system	**94.26**	**94.26**	94.04	**94.48**	1.00000
Palkovskii and Belov [23]	**90.80**	**90.78**	92.76	88.92	1.00027
Alaa Saleh Altheneyan *et al.* (PlagLinSVM) [11]	90.15	90.01	89.75	90.55	1.00210
Oberreuter and Eiselt [24]	89.30	89.27	87.17	91.54	1.00051
Sanchez-Perez *et al.* [25]	89.21	89.20	86.61	**91.98**	1.00026
Glinos [26]	89.89	88.77	**96.01**	84.51	1.01761
Alaa Saleh Altheneyan *et al.* (PlagRbfSVM) [11]	88.40	88.27	85.52	91.49	1.00209
Shrestha *et al.* [27]	87.05	86.81	84.42	89.84	1.00381
Gross and Modaresi [28]	86.84	85.50	92.52	81.82	1.02187
Rodríguez Torrejón and Martín Ramos [29]	84.87	84.87	90.03	80.27	1.00000

* The best results are highlighted in bold.

When analyzing our system output, we found that most of the incorrect results are due to the following situations:

- Sentential redundancy: This situation occurs when the sentence near the plagiarism passage is semantically related to the plagiarism passage. In that case, the system often includes it to the plagiarism passage.
- Word missing or redundancy: This situation occurs when only a part of the sentence is in the plagiarism passage. The pre-processing step has removed stopwords from the input documents. Therefore, when restoring the original text from the output of the word-phase, we need to recover these stopwords from the original documents. Redundance or some missing stopwords may occur at the beginning and the end of the recovery passage.

These problems will be considered in our future work.

4. CONCLUSION

This paper has proposed an approach using feature extraction techniques and a two-phase plagiarism detection system based on multi-layer LSTM Networks to determine plagiarism strings between two documents. The key to the paper's success is to select appropriate features for both word matching and semantic-based plagiarism. Besides, the inheritance of research results on measuring the similarity between two sentences is also an essential factor in catching sentences inside plagiarism passages compared to outside sentences. The proposed method was evaluated using the PAN 2014 text alignment corpus and widely accepted evaluation metrics: precision, recall, and plagdet. The solution achieves the best recall and plagdet, and the second precision better compared to state-of-the-art systems. In our future work, we plan to find a method to automatically choose optimal parameters for our system. Also, we will investigate methods to solve the redundancy problem in the system' output, mentioned in section 3.2.

REFERENCES

[1] A. Barrón-Cedeño, M. Vila, M. Antònia Martí, and P. Rosso, "Plagiarism meets paraphrasing: Insights for the next generation in automatic plagiarism detection," *Computational Linguistics*, vol. 39, no. 4, pp. 917-947, 2013, doi: 10.1162/COLI_a_00153.

[2] S. M. Alzahrani, N. Salim, and A. Abraham, "Understanding plagiarism linguistic patterns, textual features, and detection methods," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 2, pp. 133-149, 2012, doi: 10.1109/TSMCC.2011.2134847.

[3] Y. Liu, C. Sun, L. Lin, X. Wang, and Y. Zhao, "Computing semantic text similarity using rich features," *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, 2015, pp. 44-52, Available: http:// https://www.aclweb.org/anthology/Y15-1006.pdf.

[4] C. Grozea, C. Gehl, and M. Popescu, "ENCOPLOT: Pairwise sequence matching in linear time applied to plagiarism detection," *3rd PAN Workshop. Uncovering Plagiarism, Authorship and Social Softwar*e Misuse, 2009, Available: http://ceur-ws.org/Vol-502/paper2.pdf.

[5] M. Elhadi and A. Al-Tobi, "Duplicate detection in documents and webpages using improved longest common subsequence and documents syntactical structures," *2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*, Seoul, Korea (South), 2009, pp. 679-684, doi: 10.1109/ICCIT.2009.235, doi: 10.1109/iccit.2009.235.
[6] J. Kasprzak, M. Brandejs, and M. Kripac, "Finding plagiarism by evaluating document similarities," *Proceedings of the SEPLN'09 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse*, vol. 9, no. 4. Pp. 24-28, 2009, Available: http://ceur-ws.org/Vol-502/paper4.pdf.
[7] A. Barrón-Cedeño, C. Basile, M. D. Esposti, and P. Rosso, "Word length n-grams for text re-use detection," *International Conference on Intelligent Text Processing and Computational Linguistics- CICLing 2010*, vol. 6008, 2010, pp. 687-699, doi: 10.1007/978-3-642-12116-6_58.
[8] M. Murugesan, W. Jiang, C. Clifton, L. Si, and J. Vaidya, "Efficient privacy-preserving similar document detection," *The VLDB Journal*, vol. 19, no. 4, pp. 457-475, 2010, doi: 10.1109/GLOCOM.2017.8254012.
[9] E. Gharavi, H. Veisi, K. Bijari, and K. Zahirnia, "A fast multi-level plagiarism detection method based on document embedding representation," *Forum for Information Retrieval Evaluation- FIRE 2016*, vol. 10478, 2016, pp. 94-108, doi: 10.1007/978-3-319-73606-8_7.
[10] H. Cherroun and A. Alshehri, "Disguised plagiarism detection in Arabic text documents," *2018 2nd International Conference on Natural Language and Speech Processing (ICNLSP)*, Algiers, Algeria, 2018, pp. 1-6, doi: 10.1109/ICNLSP.2018.8374395.
[11] A. S. Altheneyan and M. El Bachir Menai, "Automatic plagiarism detection in obfuscated text," *Pattern Analysis and Applications*, vol. 23, pp. 1627-1650, 2020, doi: 10.1007/s10044-020-00882-9.
[12] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, Canada, 2013, pp. 6645-6649, doi: 10.1109/ICASSP.2013.6638947.
[13] S. Abnar, M. Dehghani, H. Zamani, and A. Shakery, "Expanded n-grams for semantic text alignment," *Cappellato*, pp. 928-938, 2014, Available: http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-AbnarEt2014.pdf.
[14] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using Siamese bert-networks," *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, Hong Kong, China, 2019, pp. 3982-3992, doi: 10.18653/v1/D19-1410.
[15] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *NIPS'13: Proceedings of the 26th International Conference on Neural Information Processing Systems*, vol. 2, 2013, pp. 3111-3119, Available: https://arxiv.org/abs/1310.4546.
[16] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," *The 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532-1543, doi: 10.3115/v1/D14-1162.
[17] M. Busta, L. Neumann, and J. Matas, "Fastext: Efficient unconstrained scene text detector," *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015, pp. 1206-1214, doi: 10.1109/ICCV.2015.143.
[18] A. Conneau, D. Kiela, H. Schwenk, and L. Barrault, "Supervised learning of universal sentence representations from natural language inference data," *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 670-680, doi: 10.18653/v1/D17-1070.
[19] D. Cer, *et al*., "Universal sentence encoder," *EMNLP demonstration, Association for Computational Linguistics*, Brussels, Belgium, 2018, Available: https://arxiv.org/abs/1803.11175
[20] J. Allan, C. Wade, and A. Bolivar, "Retrieval and novelty detection at the sentence level," *The 26th annual international ACM SIGIR conference on research and development in information retrieval*, 2003, pp. 314-321, doi: 10.1145/860435.860493.
[21] M. Potthast, *et al*., "Overview of the 5th international competition on plagiarism detection," *CLEF Conference on Multilingual and Multimodal Information Access Evaluation*, 2013, Available: http://ceur-ws.org/Vol-1179/CLEF2013wn-PAN-PotthastEt2013.pdf.
[22] M. Potthast, *et al*., "Overview of the 6th International Competition on Plagiarism Detection," *CLEF 2013 Evaluation Labs and Workshop*, Cappellato, pp. 845-876, 2013, Available: http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-PotthastEt2014.pdf.
[23] Y. Palkovskii and A. Belov, "Developing high-resolution universal multi-type n-gram plagiarism detector," *Conference and Labs of the Evaluation Forum and Workshop (CLEF'14)*, 2014, pp. 984-989, Available: http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-PalkovskiiEt2014.pdf.
[24] G. Oberreuter and A. Eiselt, "Submission to the 6th international competition on plagiarism detection, From Innovand. io, Chile," 2014, Available: https://www.uni-weimar.de/medien/webis/events/pan-14/pan14-web/
[25] M. A. Sanchez-Perez, G. Sidorov, and A. F. Gelbukh, "A Winning Approach to Text Alignment for Text Reuse Detection at PAN 2014," CLEF (Working Notes), 2014, Available: http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-SanchezPerezEt2014.pdf.
[26] D. G. Glinos, "A Hybrid Architecture for Plagiarism Detection," CLEF (working notes). 2014, Available: http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-Glinos2014.pdf.
[27] P. Shrestha, S. Maharjan, and T. Solorio, "Machine Translation Evaluation Metric for Text Alignment," CLEF (working notes), 2014, Available: https://pan.webis.de/downloads/publications/papers/shrestha_2014.pdf.
[28] P. Gross and P. Modaresi, "Plagiarism Alignment Detection by Merging Context Seeds," CLEF (working notes), 2014, Available: https://pan.webis.de/downloads/publications/papers/gross_2014.pdf

[29] D. A. Rodríguez Torrejón and J. M. Martín Ramos, "Coremo 2.3 plagiarism detector text alignment module," Notebook for PAN at CLEF (2014), Available: http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-RodriguezTorrejonEt2014.pdf

BIOGRAPHIES OF AUTHORS

Nguyen Van Son is a PhD student at MIST, Vietnam. He is now a software developer at Institute of Information Technology, MIST. He is currently a researcher at Institute of Information Technology, MIST. His research interests include machine learning and NLP.

Le Thanh Huong received her B.Eng. degree in Information Technology from Hanoi University of Science and Technology in 1997, Master degree in Computer Science from Free University of Brussels, Belgium in 2000. She received her Ph.D. in Natural Language Processing from Middlesex University, United Kingdom in 2004. She has been an Associate Professor in Computer Science since 2012. She is currently a lecturer at School of Information and Communication Technology, Hanoi University of Science and Technology. Her research interests include artificial intelligence, concentrating on natural language processing and machine learning.

Nguyen Chi Thanh received B.Eng. degree in Information Technology from Hanoi University of Science and Technology in 2003, M.Eng. in Management and Information Systems Engineering from Nagaoka University of Technology, Japan in 2008. He received Ph.D. in Information Science and Control Engineering from Nagaoka University of Technology, Japan in 2012. He is currently a researcher at Institute of Information Technology, MIST. His research interests include machine learning, NLP, and computer vision.