

Developing a Chatbot for Important Message Detection in LINE Messenger Using KMP Algorithm

Taufiq Husada Daryanto / 13518058

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jalan Ganesha 10 Bandung

taufiqhd123@gmail.com

Abstract—In this day and age, social media is growing rapidly. Almost all people in this world use social media in their everyday life. In the diverse kind of social media, LINE Messenger is one of the mostly used social media today. One of the features from LINE Messenger is group chatting. Using group chat, people can send message to a group of people effectively. But many times, communication in a group chat become ineffective because the number of unimportant messages in group chat. The presence of many unimportant messages in the group chat will make important messages will less be seen by people on the group. With that said, we need a solution to detect whether a message in chat group is important or not. In this paper, one solution of detecting important message in LINE Messenger will be discussed which is by implementing a chatbot to detect important message using KMP algorithm.

Keywords—important; chatbot; detection; KMP algorithm; string matching

I. INTRODUCTION

In today's technological era, the use of social media is massive, almost all of people in this world use social media. This phenomenon is reasonable, considering many benefits that people can get from social media. By using social media, people can easily communicate with the others anytime and anywhere, moreover we can get all information through social media in real time.

Along with technology evolution, social media is also growing. Today, not only there are diverse type of social media, but also there are diverse features from for each social media itself. One of the features that people mostly use in social media is group chatting, which is people send messages to a group so that all people in that group can read or reply the chat. Using a group chat, people can effectively communicate to a group of people.

The usage of group chatting sometimes not quite right, for example sending messages that do not fit the context in the group or sending spam message to group chat. Those inappropriate of use of social media can lead into ineffective communication. Because if there is a lot spam and

inappropriate messages in a group chat, people in that group tend not to read all the messages in the group which it can lead to skipping over important messages.

With that said, there is an urgency for an effective method to automatically detect important message in a chat. In this paper, we will be providing one such method by developing a chatbot for important message detection in LINE Messenger using KMP algorithm. We will use LINE because it is one of the most popular social media.

II. THEORETICAL FRAMEWORK

A. String Matching Algorithm

String matching algorithm is an algorithm that check if one string is found within a larger string. The simplest algorithm for string matching is the Naïve String-Matching Algorithm or as known as brute force method. The pseudocode is as follows.

```
NAIVE-STRING-MATCHER(T, P)
1  n = T.length
2  m = P.length
3  for s = 0 to n - m
4      if P[1..m] == T[s + 1..s + m]
5          print "Pattem occurs with shift" s
```

Image 1 (Source: <https://algo.ics.hawaii.edu/~nodari/teaching/s15/Notes/Topic-23.html>)

Beside the naïve approach, there are several algorithms for string matching, for example Rabin-Karp-Algorithm, Finite Automata, Knuth-Morris-Pratt Algorithm, and Boyer-Moore Algorithm [1]. The comparison of string-matching algorithms is as follows.

Algorithm	Preprocessing time	Matching time
Naive	0	$O((n - m + 1)m)$
Rabin-Karp	$\Theta(m)$	$O((n - m + 1)m)$
Finite automaton	$O(m \Sigma)$	$\Theta(n)$
Knuth-Morris-Pratt	$\Theta(m)$	$\Theta(n)$

Image 2 (Source: <https://algo.ics.hawaii.edu/~nodari/teaching/s15/Notes/Topic-23.html>)

B. Applications of String Matching Algorithm

String matching algorithm is already applied in various real-world problems, for example as follows [2].

1. **Spelling checkers:** Check the spelling of a word is correct or not
2. **Spam detection:** Identify a message is a spam or not
3. **Search engines:** Search a content based on keywords in internet
4. **Plagiarism detection:** Identify the level of similarities between two text and classify those as plagiarism case or not
5. **DNA sequencing:** Search the pattern of amino acid sequences in DNA data.

C. Knuth–Morris–Pratt Algorithm

Knuth–Morris–Pratt (KMP) algorithm is one of the string-matching algorithms. The basic idea of KMP algorithm is look for the pattern in the text from left to right, but whenever mismatch happen, it shifts the pattern more intelligently than the brute force algorithm [2]. In order to shifts the pattern more intelligently, this algorithm uses the information of the largest prefix that is also a suffix (LPS) in the pattern.

The pseudocode of KMP algorithm based on the reference [3] is as follows.

```

algorithm KMP ( $P[0, \dots, m-1]$ ,  $T[0, \dots, n-1]$ )
input: pattern P of length m and text T of length n
preconditions:  $0 \leq m-1 \leq n-1$ 
output: true if P appears in T, false if not appears

j ← 0;
i ← 0;
while (i < n) {
  if (P[j] == T[i]) {
    if (j == m-1) {
      return true;
    }
    j ← j + 1;
    i ← i + 1;
  }
  else
  {
    if (j > 0) { j ← LPS[j-1] }
    else { i ← i+1 }
  }
}
return false;
}

```

algorithm Compute-LPS-values($P[0, \dots, m-1]$)

input: pattern P of length m
preconditions: $1 \leq m$
output: table LPS[0, ..., m-1]

```

create table LPS[0, ..., m-1]
LPS[0] ← 0;
j ← 0;
i ← 1;
while (i < m) {
  if (P[j] == P[i]) {
    LPS[i] ← j+1;
    j ← j + 1;
    i ← i + 1;
  }
  else
  {
    if (j > 0) { j ← LPS[j-1] }
    else {
      LPS[i] ← 0;
      i ← i+1
    }
  }
}
}

```

The time complexity for KMP algorithm is as follows

- $O(m)$ to compute LPS values
- $O(n)$ to compute the string matching
- The total time complexity is $O(n + m)$

D. Chatbot

Chatbot is a computer program that simulates and processes human conversation that allows humans to interact with digital devices as if they were communicating with a real person [4]. Chatbot capabilities vary, from a chatbot that can only give simple response to a chat, to a chatbot that can act as a digital assistant that can give complex responses.

Today, chatbots are used in various aspects, such as chatbots for customer support, chatbots for product suggestions, chatbots for personal digital assistant, and many more. The massive usage of chatbot is because of its benefits, for example in business chatbot can give higher customer satisfaction and saving time. Furthermore, right now this chatbot technology is still growing [5].

E. API

API or application programming interface is a set of routines, protocols, and tools for building software applications. Basically, an API specifies how software components should interact [6].

There are several types of API, for example Google Maps API which let developers embed Google Maps on webpages, YouTube APIs, etc.

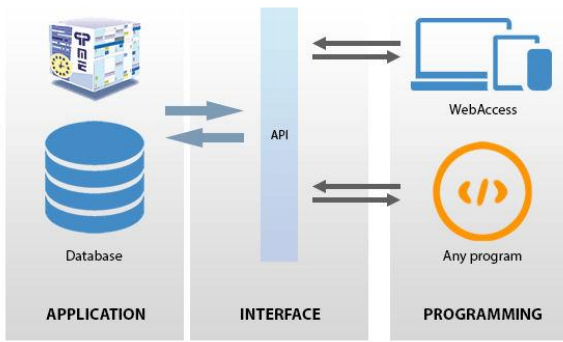


Image 3 API Illustration (source: <https://www.planningpme.com/planningpme-api.htm>)

F. LINE Messaging API

LINE Messaging API is an API that created by LINE that is usually used in building chatbots. The way it works is by allowing data to be passed between our bot server and the LINE platform [7]. The process is as follows.

1. The user sends a message to the LINE Official Account.
2. The LINE Platform sends a webhook event to the webhook URL of the bot server
3. According to the webhook event, the bot server responds to the user through the LINE Platform.

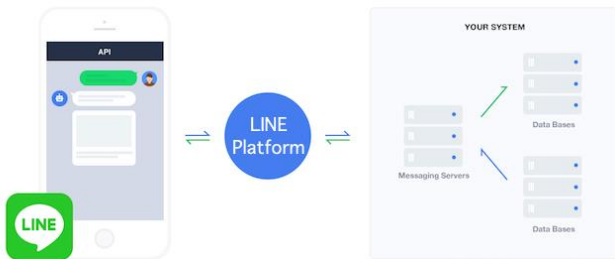


Image 4 LINE Messaging API process (source: <https://developers.line.biz/en/docs/messaging-api/overview/#what-you-can-do>)

Using this messaging API, developers can create a LINE chatbot that capable of several features, for example as follows.

- Send automatic reply messages
- Send push messages
- Send various message
- Get content sent by users
- Get user profiles
- Join group chats

- Use rich menus
- Use beacons
- Use account link
- Get the number of sent messages

III. IMPLEMENTATION

The basic idea of our chatbot implementation is we shall first defined the list of important keywords based on our preferences, then whenever the bot get a message, the bot will check if there are some words (one or more than one) from our list of important keywords that are found in the message. If found, then the bot will categorize that message as important message. In order to check whether a keyword appears in the message, we will use KMP algorithm.

The detailed explanation of the implementation is as follows:

A. KMP Algorithm Implementation for String Matching

The implementation of the KMP algorithm for string matching using python is as follows

```
def kmp_matcher(text, pattern):
    len_text = len(text)
    len_pattern = len(pattern)

    lps = compute_lps(pattern)

    i = 0
    j = 0

    while(i < len_text):
        if (text[i].lower() == pattern[j].lower()):
            if (j == len_pattern-1):
                return True
            j = 0
        else:
            j = j+1
            i = i+1
        elif (j > 0):
            j = lps[j-1]
        else:
            i = i+1

    return False
```

```

def compute_lps(pattern):
    len_pattern = len(pattern)
    lps = [0 for i in range(len_pattern)]
    i = 1
    j = 0
    while(i < len_pattern):
        if (pattern[i].lower() == pattern[j].lower()):
            lps[i] = j+1
            i+=1
            j+=1
        elif (j > 0):
            j = lps[j-1]
        else:
            lps[i] = 0
            i+=1

    return lps

```

B. Important Message Detection Implementation

First, we create the list of important keywords, then for each word in that list, we check if it is found in the message or not.

The implementation is as follows.

```

list_important_keyword = ["tugas", "ujian", "kumpul",
"taufiq", "emergency", "penting", "gawat"]

def solve(message_line):
    for keyword in list_important_keyword:
        if (kmp_matcher(message_line, keyword)):
            return True
    return False

```

C. Chatbot Implementation

The overview of the chatbot we created is that if someone send a message to a LINE group chat, then the bot will classify whether the message is important or not using Important Message Detection Algorithm that we implement before. If the message is important, then the bot will reply the message with the format as follows

```

[INFO BOT: PENTING]

/* text sent */

```

So, if people in that group want to search for the important message, they can just go to search menu in LINE chat group, then search for "INFO BOT: PENTING", and

LINE will show all message with word "INFO BOT: PENTING".

In order to build the bot, we use Flask framework and LINE messaging API. In this paper, we only show the most important part of the code which is the code for sending reply message. The code is as follows.

```

@handler.add(MessageEvent, message=TextMessage)
def handle_text_message(event):
    isImportant = solve(event.message.text)
    if (isImportant):
        line_bot_api.reply_message(
            event.reply_token,
            TextSendMessage(text="[INFO BOT:
PENTING]\n"+event.message.text))

```

IV. EXPERIMENT

We try several experiments to our chatbot by sending some messages in a group chat, and the result as is follows.

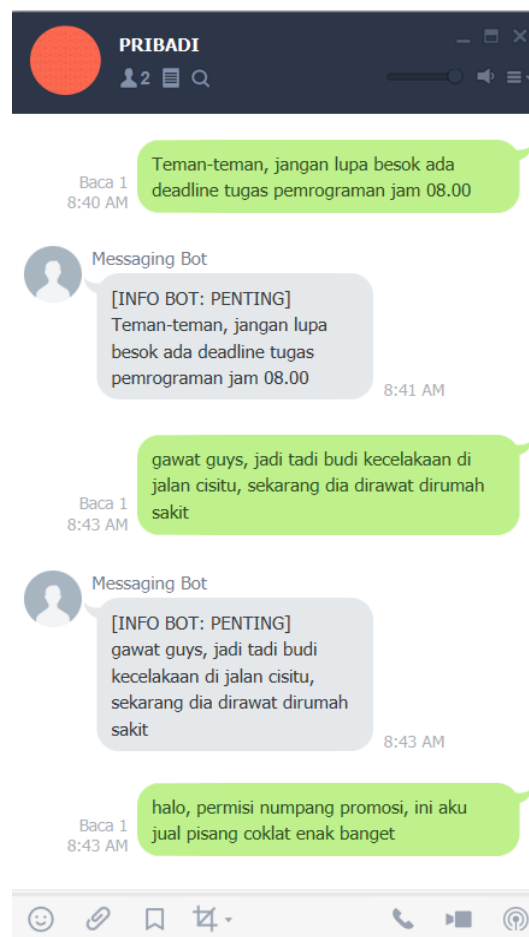


Image 5 Chatbot Experiment Result

From that experiment, we can see that there are two messages that classified as important message, which indicated

by the reply message that bot send, and there is one message that classified as unimportant message. The explanation is as follows

- The message “Teman-teman, jangan lupa besok ada deadline tugas pemrograman jam 08.00” is classified as important because there is a keyword “tugas” which is included in the list of important keywords that we created.
- The message “gawat guys, jadi tadi budi kecelakaan di jalan cisu, sekarang dia dirawat dirumah sakit” is classified as important because there is a keyword “gawat” which is included in the list of important keywords that we created.
- The message “halo, permisi numpang promosi, ini aku jual pisang coklat enak banget” is classified as unimportant message because there are no important keywords found in that message.

V. ANALYSIS

From the section IV, we have shown that our bot can work correctly. But there are several weaknesses from our bot as follows.

1. The list of important keywords is only based on personal preferences

Because the list of important keywords is only based on personal preferences, then the bot cannot be effectively used by numbers of people, because several people may have different list of important keywords

2. To modify the list of important keywords we have to change it in the code.

We store the list of important keywords in the array inside the code. So, if we want to modify the list we have to change it in the code. This makes the process less practical

3. KMP Algorithm tends to be ineffective if the size of the alphabets increases

KMP algorithm does not work so well as the size of the alphabets increases. Because there are more chances of mismatch occurs [8].

VI. IDEAS FOR IMPROVEMENTS

To improve this chatbot we have several ideas for future improvements as follows.

1. Make the chatbot give push message to user directly instead of just replying in the group chat

By making the chatbot give push message to user directly, it will make the process more practical, because the user will get important messages directly

instead of opening the group chat to see the important messages themselves.

2. Make the chatbot to be able to modify the list of important keywords directly from chat

To make the process more practical, it is better to make the bot to be able to change the list of important keywords directly from users by only using message with certain format

Right now, we have not been able to apply those improvements yet, because of time limitation. But we hope that readers can try to apply those improvements.

VII. CONCLUSION

One of the main problems from LINE group chatting is sometimes there are many unimportant messages so that important messages will less be seen. To solve that problem, we create a chatbot in LINE messenger to detect important message using KMP algorithm. From our experiment, our chatbot can effectively classify a message as important message or not. Although with that said, there still needs to be more improvements on this solution.

VIDEO LINK AT YOUTUBE

<https://www.youtube.com/watch?v=Lr9zWMDMxYk>

ACKNOWLEDGMENT

I would like to thank all the lecturers of the knowledge that is shared regarding algorithm strategies, especially Mrs. Masayu Leylia Khodra as my IF2211 algorithm strategies class lecturer. This subject given me the chance to explore more regarding several topics in the subject and its application as well as given me the chance to practice my English writing skill. I would also like to thank my friends and families who support me in the process of learning and also making this paper.

REFERENCES

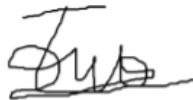
- [1] Javatpoint, “String Matching Introduction”, accessed from <https://www.javatpoint.com/> on 24 April 2020
- [2] Kapil Kumar Soni, Rohit Vyas, Amit Sinhal, “Importance of String Matching in Real World Problems”, International Journal Of Engineering And Computer Science ISSN: 2319-7242 Volume 3 Issue 6 June, 2014 Page No. 6371-6375
- [3] Rinaldi Munir, “Pencocokan String”, accessed from <https://informatika.stei.itb.ac.id/~rinaldi.munir/> on 24 April 2020
- [4] Oracle, “What Is a Chatbot?”, accessed from <https://www.oracle.com/> on 24 April 2020
- [5] Robert Williams, “Study: Global chatbot market to grow 24.4% in next 4 years”, accessed from <https://www.mobilemarketer.com/> on 24 April 2020

- [6] Vangie Beal, "API - application program interface", accessed from <https://www.webopedia.com/> on 24 April 2020
- [7] LINE Developers, "Messaging API Overview", accessed from <https://developers.line.biz/> on 24 April 2020
- [8] Kranthi Kumar Mandumula, "Knuth-Morris-Pratt Algorithm", accessed from <http://cs.indstate.edu/~kmandumula/presentation.pdf> on 24 April 2020

STATEMENT

I hereby declare that the paper I wrote is my own writing, not an adaptation, or a translation of someone else's paper, and not plagiarism.

Bandung, 24 April 2020



Taufiq Husada Daryanto 13518058