IJCCS (Indonesian Journal of Computing and Cybernetics Systems)
Vol.14, No.1, January 2020, pp. 91~102
ISSN (print): 1978-1520, ISSN (online): 2460-7258
DOI: https://doi.org/10.22146/ijccs.41289

Chatbot in Bahasa Indonesia Using NLP to Provide Banking Information

Abidah Elcholiqi*[1], Aina Musdholifah[2]
[1]Master Program of Computer Science; FMIPA UGM, Yogyakarta, Indonesia
[2]Department of Computer Science and Electronics, FMIPA UGM, Yogyakarta, Indonesia
e-mail: ***[1]abidah.elcholiqi@mail.ugm.ac.id**, [2]aina_m@ugm.ac.id

Abstrak

FAQ biasa disediakan pada website perusahaan untuk menginformasikan layanan dan produknya. Hanya saja FAQ biasanya kurang interaktif dan praktis. Chatbot dapat digunakan sebagai salah satu alternatif dalam menyediakan FAQ. Pada penelitian ini, chatbot dikembangkan untuk BTPN dalam menyediakan informasi tentang produknya yaitu Jenius. Chatbot yang dikembangkan memanfaatkan natural language processing agar sistem dapat memahami query pengguna dalam bentuk bahasa natural. Algoritma cosine similarity digunakan untuk mencari kemiripan antara query dengan pola-pola yang ada pada knowledge base. Pola dengan nilai cosine tertinggi dianggap paling mirip dengan query pengguna sehingga dapat dipakai sebagai respon untuk query pengguna. Hanya saja, algoritma ini tidak memperhatikan struktur kalimat sehingga ditambahkan pengecekan struktur kalimat dengan parse tree untuk memberi bobot pola. Dari hasil pengujian aplikasi chatbot kepada 10 penguji, didapatkan hasil tingkat kesesuaian jawaban dengan masukan pengguna sebesar 84%. Oleh karena itu chatbot yang dikembangkan dapat digunakan oleh BTPN untuk menyediakan informasi produk Jenius kepada konsumen dengan lebih interaktif dan praktis.

Kata kunci*—chatbot, natural language processing, cosine similarity, parse tree*

Abstract

FAQs are mostly provided on the company's website to inform their service and product. It's just that the FAQ is usually less interactive and presents too much information that is less practical. Chatbot can be used as an alternative in providing FAQ. In this study, chatbots were developed for BTPN in providing information about their products, namely Jenius. Chatbot developed utilizes natural language processing so that the system can understand user queries in the form of natural language. The cosine similarity algorithm is used to find similarities between queries and patterns in the knowledge base. Patterns with the highest cosine values are considered to be most similar to user queries so they can be used as a response to user queries. It's just that, this algorithm does not pay attention to the structure of the sentence so that it adds checking the structure of the sentence with the parse tree to give weight to the pattern. This chatbot application has been tested by 10 users and it was found that the suitability of the answers with user input was 84%. Therefore the chatbot developed can be used by BTPN to provide Jenius product information to consumers more interactively and practically.

Keywords—*chatbot, natural language processing, cosine similarity, parse tree*

Received November 24th,2018; Revised Januay 30th, 2020; Accepted January 31th, 2020

1. INTRODUCTION

The world wide web technology has grown rapidly allowing a revolution in terms of information exchange. One of them is the provision of FAQs on the company's website. FAQs usually display information in the form of questions and answers about the services and products that the company has. It's just that, the FAQ usually contains too much information because it covers all product information in detail. This makes access to information less interactive and practical. In addition, users sometimes have to access all pages to get the information they need, which is, it takes time.

One alternative made by the company is to provide online question and answer services, like chatbox. However, this kind of feature requires more employees to answer each user's question. The problem arises when the number of users who want to get information is very large, but the limited number of employees causes many user questions to be missed. So that automation is needed in answering user questions regarding the information needed. Therefore, chatbots can be used as an alternative to overcome this problem.

Chatbot has been widely used in many ways, including in providing entertainment, education, tourism, and so on. Chatbot can be used as a tool for learning new languages, tools for accessing information systems, tools for visualizing corpus content, and tools for answering questions on certain domains and can be trained in different languages.

In the banking sector, several studies were conducted to develop chatbots in providing bank information. One chatbot was developed in banking using natural language processing [1][2][3]. The dataset used is the FAQ data obtained from a banking website in India. The application uses rule-based and pattern-based techniques where NLP is used to process user queries. Chatbot is also used in other fields, such as providing tutors for students [4], counseling services [5], modular knowledge services [6], providers of humor [7], and so on.

In the implementation of chatbot, so that the system can respond to user queries more dynamically, the use of natural language processing plays an important role, namely in understanding user queries in natural languages. Therefore, it is necessary to use an algorithm to find query proximity with patterns in the database, such as the cosine similarity algorithm. Many cosine similarity algorithms are used to find the value of proximity between documents [8][9]. It's just that this algorithm does not consider the position of tokens in the sentence so that patterns with different sentence structures can have the same cosine value. Studies to check the structure of language were previously conducted to see the structure of the sentence[10][11]. In this paper, the cosine algorithm will be added by checking sentence structure.

BTPN is one of the companies that provides a FAQ on its website that contains products owned, namely Jenius. Jenius is a digital banking application that can be enjoyed through smartphone devices. At present, to access FAQ information, customers must move from Jenius application to browser. This is considered less practical. In addition, users sometimes have to queue when they as the customer service by chat. For this reason, it is necessary to provide automatic product information services to users through the application of chatbots.

With the use of this chatbot, consumers can more quickly get the information needed rather than having to move to the browser to read all the information on the FAQ page which is quite long or must come to the bank and meet customer service to ask for the information needed. Thus, the system only displays what information the user needs.

This paper tries to contribute to the development of chatbots in presenting information about banking products which were previously presented in the form of a FAQ system to be more interactive and practical. The built-in chatbot supports Bahasa Indonesia in reading user queries and displays responses that are more in line with user expectations.

2. METHODS

2. 1 Knowledge Base Preparation

In this research, the data used as a knowledge base is obtained from the FAQ on the BTPN website about the Jenius application. The data needs to be processed before it can be used to respond to user queries. It is also intended to increase system performance because the system does not need to pre-process the data against patterns in the database. The FAQ data used includes one of the Jenius features, namely Dreamsaver. The FAQ data consists of 20 sentence questions and answers about the Dreamsaver feature. Data is stored in files with the extension .aiml and preprocessed using Natural Language Processing (NLP). Preprocessing with NLP conducted includes:

1. *Tokenisation*
   User queries will be identified and broken down into tokens. At this stage, punctuation such as dots, commas, question marks will be omitted.
2. *Slang Word Checking*
   Tokens and then check the slang dictionary to see whether the word is a slang word and whether it can be replaced into a more standard Indonesian word or can be ignored, for example, the slang word for Bahasa Indonesia: sih, dong, deh, and so on.
3. *Morphology Checking*
   Tokens are analyzed morphologically to get the basic words and types of words. Morphological examination is carried out by removing the prefix, suffix, prefixes, and repetitions as shown in Figure 1 to get the basic word from each token.

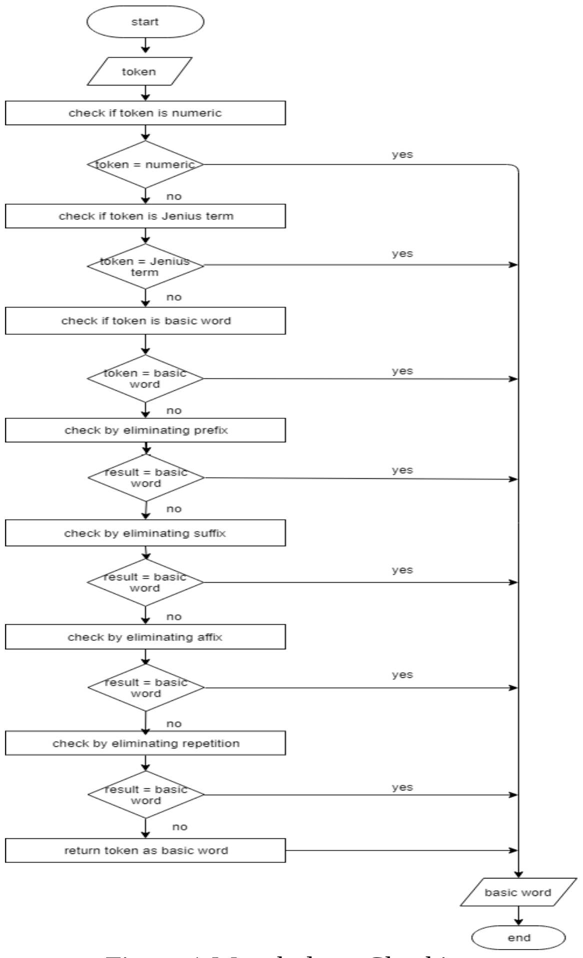
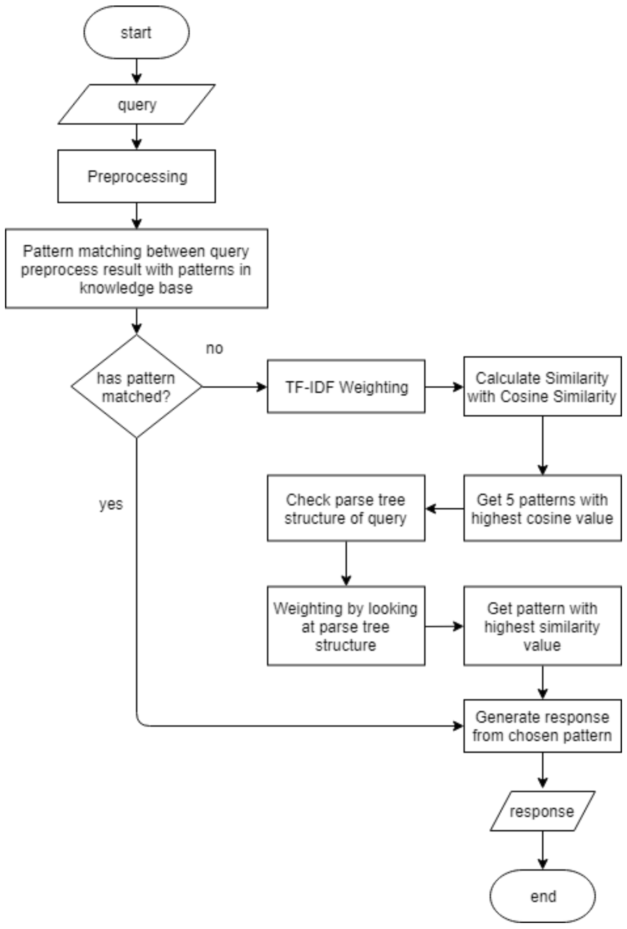

start
token
check if token is numeric
token = numeric
yes
no
check if token is Jenius term
token = Jenius term
yes
no
check if token is basic word
token = basic word
yes
no
check by eliminating prefix
result = basic word
yes
no
check by eliminating suffix
result = basic word
yes
no
check by eliminating affix
result = basic word
yes
no
check by eliminating repetition
result = basic word
yes
no
return token as basic word
basic word
end

Figure 1 Morphology Checking

2. 2 User Query Processing

The process through which the query is passed until the response is seen is shown in Figure 2. In Figure 2, the query is carried out by preprocessing first, then a pattern search is performed which is most similar to the query using the cosine similarity and parse tree algorithms.


start
query
Preprocessing
Pattern matching between query preprocess result with patterns in knowledge base
has pattern matched?
no
yes
TF-IDF Weighting
Calculate Similarity with Cosine Similarity
Get 5 patterns with highest cosine value
Check parse tree structure of query
Weighting by looking at parse tree structure
Get pattern with highest similarity value
Generate response from chosen pattern
response
end
Figure 2 User Query Processing Flow

2. 2.1 Query Preprocessing

The preprocessing stages for user query is the same as in the knowledge base preparation stage, the user query will be performed: tokenisation, slang word checking and morphological checking. For example, the following query:

user*: Gimana sih cara bikin akun Jenius?*

It will follow the tokenization process to: *gimana, sih, cara, bikin, akun, jenius*. These tokens will be checked in the dictionary slang and omitted or replaced with a more standard word if it is a slang word. After that the basic word and type of words are searched through checking morphology. From this preprocess, preprocess results are obtained: *bagaimana (WH) cara (NN) buat (VB) akun (NN) jenius (NN)*.

2. 2.2 Pattern Matching

The results of the query preprocessing are then matched with the patterns stored in the knowledge base. If the exact pattern found with the results of the pre-process query is found, the system will generate a response according to that pattern. But if there is no similar pattern, then the process is followed by a search pattern that is most similar to the cosine similarity algorithm.

2. 2.3 Similarity Pattern Seacrh with Cosine Similarity

In this paper, matching the pattern similarity using TF-IDF weighting and Cosine Similarity. In this process, patterns containing tokens that are present in the preprocess results will be included in the TF-IDF weighting process. All tokens that are in the preprocess results and patterns in the knowledge base will be calculated TF and IDF for each token by following equations (1) and (2) as follows:

$$idf_i = \log(\frac{N}{df_i}) \qquad (1)$$

Notes:

idf_i	= Inverse Document Frequency for term-*i*
N	= the number of documents to compare
df_i	= the number of documents containings term-*i*

$$w_{i,j} = tf_{i,j} \times idf_i \qquad (2)$$

Notes:

$w_{i,j}$	= documents widghting term-*i* in document-*j*
$tf_{i,j}$	= the number of frequency term-*i* in document-*j*
idf_i	= Inverse Document Frequency for term-*i*

For example, query "*bagaimana cara membuat akun Jenius*" will be compared to patterns in knowledge base as below:

K1 = Bagaimana mengisi saldo jenius

K2 = Dimana tempat membuat jenius

K3 = Bagaimana cara transfer uang

K4 = Bagaimana membuat jenius

The result of weighting TF-IDF like Table 1 below:

Table 1 Example of Weighting TF-IDF

Token/Term	Q	K1	K2	K3	K4	df	log(n/df)	Q	K1	K2	K3	K4
bagaimana	1	1	0	1	1	4	0.09691	0.09691	0.09691	0.00000	0.09691	0.09691
cara	1	0	0	1	0	2	0.39794	0.39794	0.00000	0.00000	0.39794	0.00000
buat	1	0	1	0	1	3	0.22185	0.22185	0.00000	0.22185	0.00000	0.22185
akun	1	0	0	0	0	1	0.69897	0.69897	0.00000	0.00000	0.00000	0.00000
jenius	1	1	1	0	1	4	0.09691	0.09691	0.09691	0.09691	0.00000	0.09691
isi	0	1	0	0	0	1	0.69897	0.00000	0.69897	0.00000	0.00000	0.00000
saldo	0	1	0	0	0	1	0.69897	0.00000	0.69897	0.00000	0.00000	0.00000
dimana	0	0	1	0	0	1	0.69897	0.00000	0.00000	0.69897	0.00000	0.00000
tempat	0	0	1	0	0	1	0.69897	0.00000	0.00000	0.69897	0.00000	0.00000
transfer	0	0	0	1	0	1	0.69897	0.00000	0.00000	0.00000	0.69897	0.00000
uang	0	0	0	1	0	1	0.69897	0.00000	0.00000	0.00000	0.69897	0.00000

Where the weight of the term ‘bagaimana’ to query Q = ‘*bagaimana cara membuat akun jenius*’ is:

$$w_{[bagaimana][Q]} = tf_{[bagaimana][Q]} \times idf_{[bagaimana]} = 1 \times 0.09691 = 0.09691$$

Where the weight of the term ‘bagaimana’ is also calculated against the other 4 patterns:

$w_{[bagaimana][K1]} = 1 \times 0.09691 = 0.09691$
$w_{[bagaimana][K2]} = 0 \times 0.09691 = 0$
$w_{[bagaimana][K3]} = 1 \times 0.09691 = 0.09691$
$w_{[bagaimana][K4]} = 1 \times 0.09691 = 0.09691$

The process is carried out on other tokens / terms in the same way. After obtaining the weight of each term in the pattern, then the value of proximity between the patterns is calculated using the Cosine Similarity algorithm according to equation (3) below:

$$\cos(x, y) = \frac{x.y}{\|x\| \; \|y\|} \tag{3}$$

Where . indicates the vector dot produc, $x.y = \sum_{k=1}^{n} x_k \, y_k$, and $\|x\|$ is the length of vector $x, x = \sqrt{\sum_{k=1}^{n} x_k^2} = \sqrt{x.x}$

So for the results of weighting TF-IDF as in table 1, the Cosine Similarity value can be calculated between document Q with K1, K2, K3, K4 by calculating the length of the vector of each document as follows:

$$\|Q\| = \sqrt{0.09691^2 + 0.39794^2 + 0.22185^2 + \cdots + 0^2} = 0.8455$$

Calculation results vector Q and K lengths are shown in Table 2:

Table 2 Example of Vector Length Calculation Results

Vector length				
Q	K1	K2	K3	K4
0.8455	0.9979	1.0177	1.0700	0.2608

And,

Total weight of K1 $= 0.09691 * 0.09691 + 0.39794 * 0 + \cdots + 0 * 0$
$= 0.00939 + 0 + \ldots + 0$
$= 0.01878$

The results of calculating the total weight of K with Q are shown in Table 3:

Table 3 Example of Scalar Dot Product Results

Pattern	Scalar dot product
K1	0.01878
K2	0.05861
K3	0.16775
K4	0.06800

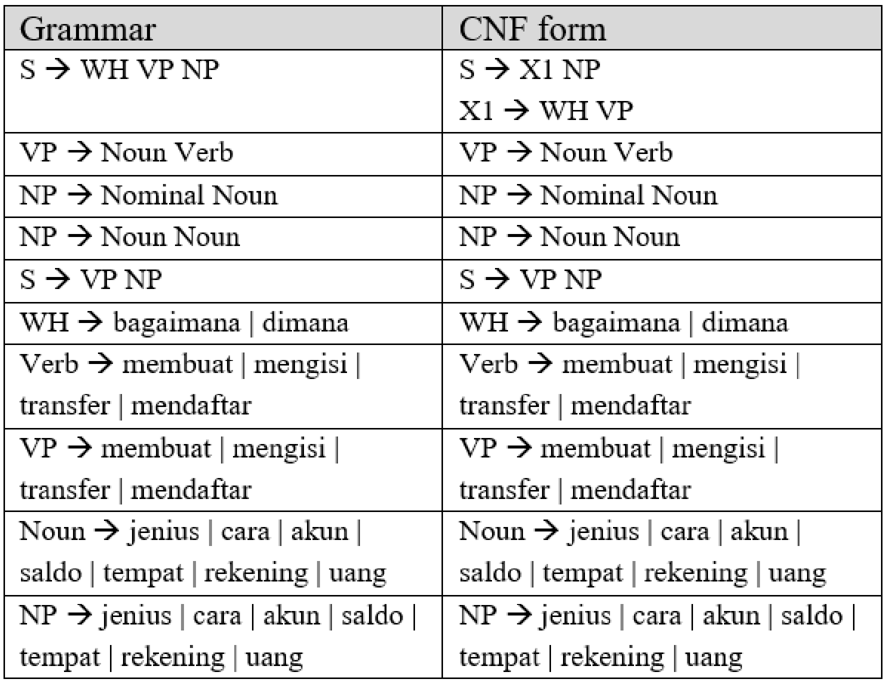
Cosine value of Q for each K documents are:

Cos (Q, K1) = 0.01878/ (0.8455*0.9979) = 0.0223
Cos (Q, K2) = 0.05861/ (08455*1.0177) = 0.0681
Cos (Q, K3) = 0.16775/ (0.8455*1.0700) = 0.1854
Cos (Q, K4) = 0.06800/ (0.8455*0.2608) = 0.3084

Of the 4 patterns compared to Q queries, K4 has the highest cosine value. However, this study added weighting with the parse tree to find the closeness value of the parsing tree structure between the query and the patterns in the knowledge base.

2. 2.4 Sentence Structure Checking with Parse Tree

Parsing on NLP is very useful in understanding the meaning of human language by looking at language grammar [11]. Parsing will form a parsing tree so we can deduce the meaning of the sentence. So in the example above, if the grammar rule follows Figure 3.



Grammar	CNF form
S → WH VP NP	S → X1 NP X1 → WH VP
VP → Noun Verb	VP → Noun Verb
NP → Nominal Noun	NP → Nominal Noun
NP → Noun Noun	NP → Noun Noun
S → VP NP	S → VP NP
WH → bagaimana | dimana	WH → bagaimana | dimana
Verb → membuat | mengisi | transfer | mendaftar	Verb → membuat | mengisi | transfer | mendaftar
VP → membuat | mengisi | transfer | mendaftar	VP → membuat | mengisi | transfer | mendaftar
Noun → jenius | cara | akun | saldo | tempat | rekening | uang	Noun → jenius | cara | akun | saldo | tempat | rekening | uang
NP → jenius | cara | akun | saldo | tempat | rekening | uang	NP → jenius | cara | akun | saldo | tempat | rekening | uang

Figure 3 Example of Grammar Rule

The results of checking the parse tree structure for Q queries are shown in Figure 4 as follows:

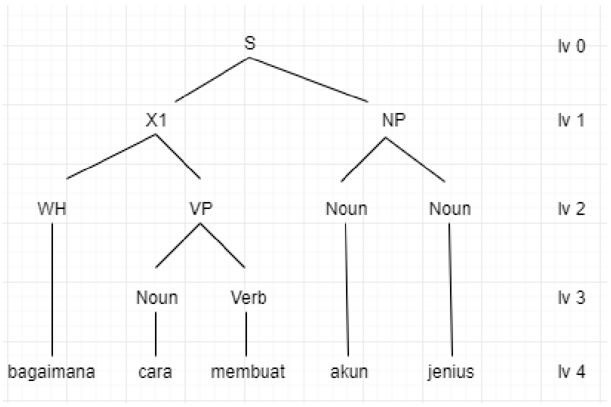


Figure 4 Example of Parse Tree of Query Q

So that the tree structure Q to K when compared is shown in Table 4:

Table 4 Example Parse Tree Structure Comparation of Q to all K

Lv	**query**	**K1**	**K2**
0 1 2 3 4	S X1 NP WH VP Noun Noun WH Noun Verb Noun Noun Bagaimana cara membuat akun jenius Number of leaf level is = **5**	S X1 NP WH VP Noun Noun Bagaimana mengisi saldo jenius	S X1 NP WH VP NP WH Noun Verb NP Dimana tempat membuat jenius
		Structure is same until level 2, w = 3/5 = 0.6	Structure is same until level 1 w = 2/5 = 0.4
		K3	**K4**
		S X1 NP WH VP NP WH Noun Verb NP Bagaimana cara transfer uang	S X1 NP WH VP NP Bagaimana membuat jenius
		Structure is same until level 1, w = 2/5 = 0.4	Structure is same until level 1, w = 2/5 = 0.4

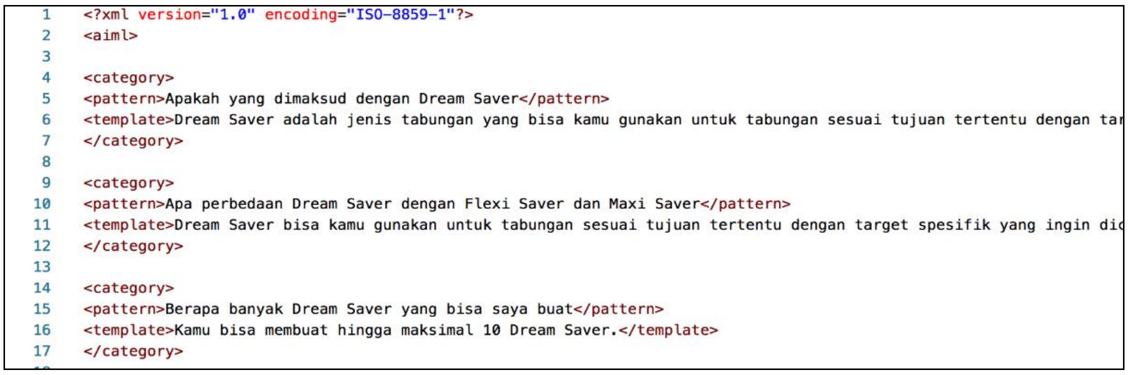
So that the calculation of the pattern similarity value is as follows:

Similarity value (Q, K1) = 0.0223 * 0.6 = 0.01338
Similarity value (Q, K2) = 0.0681 * 0.4 = 0.02724
Similarity value (Q, K3) = 0.1854 * 0.4 = 0.07416
Similarity value (Q, K4) = 0.3084 * 0.6 = 0.12336

Then, in the example query "*Bagaimana cara membuat akun jenius*", the system will display a response from the K4 pattern because it has the highest similarity value.

3. RESULTS AND DISCUSSION

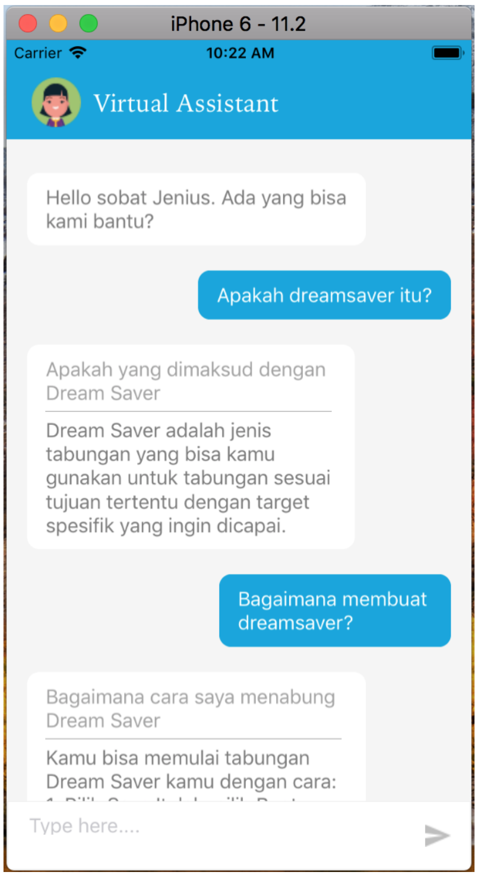
In this research, the patterns are stored in the form of a .aim file as shown in Figure 5. The pattern is processed and the preprocess results are stored in the form of a .json file.



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<aiml>

<category>
<pattern>Apakah yang dimaksud dengan Dream Saver</pattern>
<template>Dream Saver adalah jenis tabungan yang bisa kamu gunakan untuk tabungan sesuai tujuan tertentu dengan tar
</category>

<category>
<pattern>Apa perbedaan Dream Saver dengan Flexi Saver dan Maxi Saver</pattern>
<template>Dream Saver bisa kamu gunakan untuk tabungan sesuai tujuan tertentu dengan target spesifik yang ingin dic
</category>

<category>
<pattern>Berapa banyak Dream Saver yang bisa saya buat</pattern>
<template>Kamu bisa membuat hingga maksimal 10 Dream Saver.</template>
</category>
```

Figure 5 Example of Pattern Stored in Knowledge Base

Thus when a user enters a query, the system can directly use the preprocessed results stored in the knowledge base. The results of the chatbot implementation are shown in Figure 6 as follows:



Hello sobat Jenius. Ada yang bisa kami bantu?

Apakah dreamsaver itu?

Apakah yang dimaksud dengan Dream Saver

Dream Saver adalah jenis tabungan yang bisa kamu gunakan untuk tabungan sesuai tujuan tertentu dengan target spesifik yang ingin dicapai.

Bagaimana membuat dreamsaver?

Bagaimana cara saya menabung Dream Saver

Kamu bisa memulai tabungan Dream Saver kamu dengan cara:

Figure 6 Chatbot Design Implementations

The system was tested by 10 testers with the results as in table 5. From the test results, it was found that the system still had an error rate of 16% :

a. The user feels that he has entered a specific query, but the system responds to the question option so that the user selects one of these options.
b. Response is not in line with user expectations.
c. The system returns an incorrect response because the answer to the user's query is not yet available

Table 5 Chatbot Testing Results

Testers-	Total query	Total response meets expectations	Total response does not meet expectations	Is system interactive?	Is fast-response?
1	20	16	4	Yes	Yes
2	20	16	4	Yes	Yes
3	20	19	1	Yes	Yes
4	20	17	3	Yes	Enough
5	20	17	3	Yes	Yes
6	20	14	6	Yes	Yes
7	20	16	4	Yes	Yes
8	20	18	2	Yes	Yes
9	20	18	2	Yes	Enough
10	20	17	3	Yes	Yes
Percentage of response meets expectations				84 %	
Percentage of response does not meet expectations				16 %	

From user queries that are not responded to according to user expectations, such as caused by several patterns having the same cosine value and the same parse tree weight so that the system displays question options for users. In some queries also found a sentence structure that cannot be handled with grammar rules stored in the database so that the system cannot complete the calculation of the proximity of the pattern correctly. In addition, there are also user queries whose patterns are not yet available in the knowledge base, in this case it can be caused by the use of words that are not understood by the system or slang words that are not handled properly so that the results of cosine similarity calculations are not good.

In terms of system performance, testing by running as many as 28 queries on the system can be completed in 2.23 seconds with the average query being responded to less than 20 milliseconds. However, for reasons of user experience so that the chatbot is not too fast in displaying responses to users, each query on the chatbot application adds a loading time of 1-3 seconds depending on the response length. This way, user will feel like they are chatting with other human, not a computer.

4. CONCLUSIONS

Based on the research and test results, it can be concluded that the chatbot that was developed was able to provide a response to user queries with a response rate of 84%. The response mismatch of 16% is caused by the limited number of vocabulary slang and a combination of patterns with different sentence structures so that some queries produce the same value of closeness patterns, as well as answers that are expected to be unavailable. The system responds to each user query on average less than 20 milliseconds and displays a response of about 1-3 seconds after adding the waiting time depending on the response length.

5. FUTURE WORKS

Suggestions that can be used for further research, especially in research related to the development of chatbot applications is the need to enrich the dictionary of words, both Indonesian dictionaries and slang dictionaries. In addition, it is also necessary to enrich the existing patterns in the knowledge base with a combination of different patterns and sentence structures so as to increase the possibility of a better response.

REFERENCES

[1] C.S. Kulkarni, A.U. Bhavsar, S.R. Pingale, and S.S. Kumbhar, "BANK CHAT BOT – An Intelligent Assistant System Using NLP and Machine Learning," *IRJET (International Research Journal of Engineering and Technology,* vol: 04 Issue: 05 | May -2017 [Online]. Available:https://www.irjet.net/archives/V4/i5/IRJET-V4I5611.pdf [Accessed: 9-Jan-2018]

[2] S.H. Aparaj, C. Shashank, and R. Bhagat, "Smart Bot - A Virtual Help Desk Chat Bot," *ICAICTE2013* (*International Conference on Advanced Information Communication Technology in Engineering,* 2013.

[3] A. Dole, H. Sansare, R. Harekar, and S. Athalye, "Intelligent Chat Bot for Banking System," *IJETTCS (International Journal of Emerging Trends & Technology in Computer Science,* vol 4, Issue 5(2), September - October 2015 [Online]. Available: http://www.ijettcs.org/Volume4Issue5(2)/IJETTCS-2015-10-09-16.pdf [Accessed: 20-Apr-2018]

[4] R. Shah, S. Lahoti, and Lavanya, "An Intelligence Chat-bot using Natural Langauge Processing," *International Journal of Engineering Research*, vol no. 6, Issue no. 5, pp: 281-286, 2017.

[5] A. Parab, S. Palkar, S. Maurya, and S. Balpande, "An International Career Counselling Bot: A System for Counselling," *IRJET (International Journal of Engineering and Technology,* vol: 04, Issue: 03 | Mar – 2017 [Online]. Available: https://www.irjet.net/archives/V4/i3/IRJET-V4I3604.pdf [Accessed: 1-Dec-2017]

[6] B. Setiaji, E. Utami, and H.A. Fatta, "Membangun Chatbot Berbasis AIML dengan Arsitektur Berbasis Modular," *Seminar Nasional teknologi Informasi dan Multimedia 2013* STIMIK AMIKOM Yogyakarta, 2013.

[7] A. Augello, G. Saccone, and S. Gaglio, "Humorist Bot: Bringing Computational Humour in a Chat-Bot System," *International Conference on Complex, Intelligent and Software Intensive Systems,* 2018 [Online]. Available: https://ieeexplore.ieee.org/document/4606756 [Accessed: 1-Jun-2018]

[8] V.R. Prasetyo, and E. Winarko, "Rating of Indonesian Sinetron Based on Public Opinion In Twitter Using Cosine Similarity," *2nd ICST* (*International Conference on Science and Technology-Computer,* 2016 [Online]. Available: https://ieeexplore.ieee.org/document/7877374 [Accessed: 1-Jun-2018]

[9] P.P. Gokul, B.K. Akhil, and K.M. Kumar, "Sentence Similarity Detection in Malayalam Language using cosine similarity," *2nd IEEE International Conference on RTEICT (Recent Trends in Electronis Information & Communication Technology),* 2017

[Online]. Available:https://ieeexplore.ieee.org/document/8256590 [Accessed: 5-Jun-2018]

[10] R.A. Sukamto, “Penguraian Bahasa Indonesia dengan Menggunakan Pengurai Collins,” Thesis, Master Program of Informatics, Institut Teknologi Bandung, 2009.

[11] E.M. Sibarani, M. Nadial, E. Panggabean, and Meryana, “A Study of Parsing Process on Natural Language Processing in Bahasa Indonesia,” *IEEE 16th International Conference on Computational Science and Engineering*, 2013 [Online]. Available: https://ieeexplore.ieee.org/document/6755234 [Accessed: 5-Apr-2018]