Chat Conversation Segmentation Based on Question-and-Answer Patterns Using Regular Expressions

Adinda Putri - 13523071
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: adindaputri0517@gmail.com , 13523071@std.stei.itb.ac.id

***Abstract*—Text-based conversations have become an essential part of digital interaction, particularly within online forums, customer service chats, and messaging platforms. With the increasing volume of conversations, the need to manage and analyze this data is paramount. Identifying question-and-answer patterns within conversations can significantly aid in segmenting and understanding the flow of information. This research proposes the application of Regular Expressions (Regex) to identify and delineate conversation segments based on structured question-and-answer (Q&A) patterns. Through this approach, an effective method for automated conversation segmentation can be developed, which in turn can enhance the efficiency of conversation analysis, information management, and the development of automated response systems.**

Keywords—Conversation Segmentation, Question-and-Answer Patterns, Regular Expressions, Text Analysis

I. Introduction

In today's digital communication landscape, text-based conversations have become a central part of everyday interaction, spanning private messages, customer support, and online forums. The increasing volume of such conversations presents growing challenges in managing and analyzing information effectively, particularly when conversation structures are informal and unstructured.

Studies have shown that unstructured dialogues hinder automatic information extraction due to the absence of clear semantic boundaries (Serban et al., 2015). One effective strategy to address this issue is conversation segmentation based on question-and-answer (Q&A) patterns, which divides lengthy dialogue into meaningful units (Chen & Wang, 2019).

This study introduces the use of Regular Expressions (Regex) as a simple yet powerful method to identify Q&A patterns in text-based conversations. Regex offers key advantages such as computational efficiency, ease of implementation, and no requirement for machine learning model training (Jurafsky & Martin, 2020). It can detect elements like question marks, interrogative terms (e.g., "what," "who," "when," "how"), and common answer starters (e.g., "yes," "no," "I think").

With this approach, conversations can be segmented into structured Q&A pairs, making the data easier to interpret and analyze. This structured output is highly beneficial for enhancing systems such as chatbots, conversational analytics tools, and automatic archiving (Zhou et al., 2018).

The findings of this research show that Regex-based segmentation effectively identifies meaningful Q&A structures, improving the precision of information retrieval and the efficiency of conversational analysis. The method presents a scalable and practical solution for automated processing of conversational data.

II. Basic Theory

A. *Text-Based Conversations and Segmentation*

In the digital era, text-based conversations have become a primary mode of human interaction, spanning from casual messaging to customer service chats and discussions in online forums. These conversations possess unique characteristics: they are often informal, lack prosodic and non-verbal cues, and tend to be fragmented or heavily context-dependent (Jurafsky & Martin, 2020). Such traits distinguish them from spoken interactions and introduce specific challenges for computational analysis.

One of the core difficulties in processing text-based conversations lies in their unstructured nature. Without clear boundaries between questions, answers, and topical shifts, automated systems struggle to extract relevant information efficiently. As noted by Serban et al. (2015), the absence of annotated segmentation in many dialogue datasets hinders the development of robust conversational models, making it difficult for machines to interpret the flow of conversation.

To address this challenge, conversation segmentation plays a vital role. It involves dividing a conversation into smaller, coherent units that capture specific interactions or shifts in intent. A widely adopted approach is segmenting based on question-and-answer (Q&A) patterns, which has proven effective in structuring dialogue. Chen and Wang (2019) highlight that recognizing dialogue acts, such as questions, confirmations, and elaborations, is essential for understanding communication flow and intent. Well-defined Q&A pairs enhance the interpretability of dialogue for both human analysts and computational systems.

Practically, conversation segmentation supports various downstream natural language processing tasks, including information retrieval, sentiment analysis, and chatbot development. For instance, the success of XiaoIce, a socially intelligent chatbot developed by Zhou et al. (2018), demonstrates how structured dialogue enables more coherent, contextually aware, and engaging interactions. Furthermore, segmentation simplifies the learning process for dialogue systems, as it reduces ambiguity and enhances the system's ability to associate queries with appropriate responses (Serban et al., 2015).

B. Question-and-Answer (Q&A) Patterns

Question-and-answer (Q&A) patterns offer a straightforward way to structure conversational data by focusing on how a question is followed by a response. This approach is different from general segmentation methods that rely on turn-taking or time-based boundaries. Instead, Q&A segmentation looks at the relationship between utterances, where one message asks something, and another replies with relevant information. In text-based conversations, where there's no tone, gesture, or visual cue to guide interpretation, this method helps clarify the flow of interaction.

Using Q&A pairs gives structure to what is otherwise a stream of unorganized text. A question signals what the speaker wants to know, and the answer tries to address that need. When these parts are grouped together, it's easier to identify what the conversation is about, how people are responding, and whether the exchange is complete. This kind of segmentation is useful for tasks like searching through conversations, summarizing long chats, or helping automated systems generate replies.

Q&A formats also show up in a variety of real-world data, like support chats, community forums, and comment sections. Even when people write informally or skip punctuation, the general structure of asking and answering tends to remain. Systems that can spot these patterns can better highlight unresolved problems, frequently mentioned topics, or user feedback without needing to understand the entire conversation.

This pattern is especially helpful when building systems that need to understand how people talk. By learning from examples of questions and their replies, a model can start to predict what kind of answer makes sense in a given situation. This helps it respond more naturally and stay on topic during a conversation.

C. Regular Expression (Regex)

Regular expressions, commonly known as regex, are patterns used to search, match, and manipulate text based on defined rules. Regex provides a powerful way to identify and extract complex text patterns, such as email addresses, phone numbers, or specific keyword structures. A regular expression is composed of a combination of literal characters and special characters known as *metacharacters*, which together form the pattern to be matched.

Regex operates with high efficiency and is widely supported across programming languages and tools. It is especially useful for filtering and structuring text-based data without requiring extensive computational resources. There are two main components in regex syntax.

1. Literal characters

   These are standard characters that represent themselves directly, including digits (0–9), letters (a–z, A–Z), and other alphanumeric symbols. For example, the pattern book matches the exact string "book" in the text.

2. Metacharacters

   Metacharacters are symbols that carry special functions in a regex pattern, allowing it to perform more than simple literal matching. They support operations such as grouping, alternation, anchoring, and repetition.

   One category within metacharacters is quantifiers, which define how many times a given pattern should appear. This aspect is particularly useful when handling variations in phrasing and repetition within conversational text.

Table 1. Frequently used metacharacters

Construct	**Description**
.	Matches any single character except a newline
+	Matches one or more repetitions of the preceding element
*	Matches zero or more repetitions
?	Matches zero or one occurrence
^	Asserts position at the start of a line
$	Asserts position at the end of a line
[]	Defines a character class (e.g., [a-z])
()	Groups expressions
{}	Specifies repetition (e.g., {2,4} matches 2 to 4 times)
|	Acts as an OR operator

In addition to literal characters and metacharacters, regular expressions also feature a third important construct: predefined character classes. These are shorthand notations used to represent common sets of characters, allowing more efficient and readable pattern definitions.

Predefined character classes are especially useful in text-based conversations for identifying common elements such as digits, words, and spacing.

Table 2. Frequently used classes

Construct	Description
\d	Matches any digit (0–9)
\D	Matches any non-digit character
\w	Matches any word character (a–z, A–Z, 0–9, and underscore)
\W	Matches any non-word character
\s	Matches any whitespace character, such as space, tab, or newline
\S	Matches any non-whitespace character
\b	Matches a word boundary (e.g., \bhow\b matches “how”, but not “however”)
\B	Matches a position that is not a word boundary

These classes are particularly useful when identifying key tokens in conversational inputs, such as numeric values, question keywords, or named entities. By categorizing words into specific token classes, the segmentation process becomes more structured and accurate, allowing the system to better understand the role of each word within a question or response.

III. Implementation and Discussion

To demonstrate that conversation segmentation based on question- To demonstrate that conversation segmentation based on question-and-answer patterns can be done effectively, this study adopts a straightforward step-by-step approach. The process begins by preprocessing a real-world conversation dataset using Python. Next, question and answer patterns are identified using regular expressions (regex). Valid question-answer pairs are then statistically analyzed and saved.

To support this process, several Python libraries are used, pandas for data manipulation, re for regular expression operations, and ast for parsing stringified list data into Python list objects. For building a simple chatbot as a practical application of the segmented data, the study also uses the sentence_transformers library along with util from the same module to compute sentence embeddings and cosine similarity scores. These tools help match user input with semantically similar questions from the dataset, enabling the bot to provide the most relevant responses.

```python
import pandas as pd
import re
```
[70] ✓ 0.0s Python

Figure 1. Library Import
Source: Author

A. *Dataset Exploration*

This project uses the **Human Conversation Training Data** dataset, which is publicly available on Kaggle. The dataset contains a large collection of casual conversations between two participants, labeled as *Human 1* and *Human 2*. Each line in the file represents one turn in the dialogue, beginning with the speaker’s label followed by their spoken text.

The dataset is stored in plain text format, without any structured delimiters like CSV or JSON. This makes it ideal for experimenting with custom parsing techniques using tools like regular expressions. Since every line alternates between speakers, it provides a natural setup for modeling turn-based conversations.

To prepare the data for segmentation, the text file is read line by line using Python and then cleaned by removing empty lines and leading/trailing whitespaces. The cleaned lines are concatenated into a single string variable, conversation_text, which simplifies the process of splitting the dialogue into individual turns and extracting question-answer pairs for further analysis.

```python
with open('../data/human_chat.txt', 'r', encoding='latin-1') as file:
    lines = file.readlines()

conversation_text = "\n".join([line.strip() for line in lines if line.strip() != ""])
```
[71] ✓ 0.0s Python

Figure 2. Data Import
Source: Author

B. *Q&A Segmentation Using Regular Expressions*

The goal of this stage is to segment the raw conversation into structured question-and-answer (Q&A) pairs. To accomplish this, the script uses regular expressions (regex) to identify which lines contain questions and which follow as potential answers. This segmentation forms the core of the study, allowing further analysis and application, such as building a chatbot, based on these structured pairs.

The identification of questions relies on a regex pattern defined as question_pattern. This pattern matches any line that begins with common English interrogatives (e.g., *what*, *why*, *how*, *can*, *should*) and ends with a question mark. It also includes a fallback for any line that ends with a ?, to catch cases where the question form is less structured. The re.MULTILINE flag ensures the pattern can operate across multiple lines of input.

```python
question_pattern = re.compile(
    r"(?i)\b(what|where|when|why|who|how|is|are|do|does|did|can|could|will|would|should|have|has|had)\b.*?\?|"
    r".*?\?", re.MULTILINE
)
```
[72] ✓ 0.0s Python

Figure 3. Regex Pattern for Question Detection
Source: Author

Before applying this pattern, the raw conversation text is split into individual speaker turns using another regex pattern (turn_split_pattern) that identifies prefixes like “Human 1:” or “Human 2:”. These labels are then removed, and only the utterances themselves are preserved in the dialogue_turns list.

The segmentation logic is implemented in the function segment_conversation_qna(conversation_text). This function loops through each turn: when it finds a question (based on the regex), it starts a new segment. Any following lines that are not questions are collected as answers to the current question. When a new question is encountered, the previous Q&A segment is stored, and a new one begins. This continues until all turns are processed, at which point the final Q&A pair is also appended.

```python
def segment_conversation_qna(conversation_text):
    segments = []
    current_question = None
    current_answers = []

    parts = re.split(turn_split_pattern, conversation_text)
    dialogue_turns = []
    for i in range(1, len(parts), 2):
        if i + 1 < len(parts):
            utterance = parts[i+1].strip()
            dialogue_turns.append(utterance)

    for turn in dialogue_turns:
        if question_pattern.search(turn):
            if current_question and current_answers:
                segments.append({'question': current_question, 'answers': current_answers})
            current_question = turn
            current_answers = []
        else:
            if current_question:
                current_answers.append(turn)

    if current_question and current_answers:
        segments.append({'question': current_question, 'answers': current_answers})
    elif current_question:
        segments.append({'question': current_question, 'answers': []})

    return segments
```
[77] ✓ 0.0s Python

Figure 4. Q&A Segmentation Function
Source: Author

This approach ensures that each segment is centered around a clear question, with one or more associated responses, effectively structuring unstructured chat data into a format suitable for further analysis or interactive systems.

```python
segments = segment_conversation_qna(conversation_text)
df = pd.DataFrame(segments)
```
[78] ✓ 0.0s Python

Figure 5. Conversion of Segments into DataFrame
Source: Author

C. Statistical Analysis of Q&A Segments

The statistical analysis of the segmented conversation data reveals several important characteristics. A total of 394 question–answer segments were successfully extracted, indicating that the dataset contains a substantial number of distinct conversational turns structured around questions. On average, each question was followed by approximately 2.17 answers, showing that conversations often involved more than just a single direct response. However, this number varied quite a bit, as shown by the standard deviation of 1.67, some questions received only one answer, while others prompted a longer exchange.

Looking deeper, the minimum number of answers for a question is 1, while the maximum reaches up to 11. This suggests that certain questions sparked broader discussions, possibly due to their open-ended nature. The percentiles (25%, 50%, and 75%) show that most questions were answered between one and three times, confirming that while a few questions led to long discussions, the majority received just one or two responses. Overall, these statistics help validate that the segmentation approach captured both short and extended interactions in a meaningful way.

```python
print("Total Q&A pairs:", len(df))
df['num_answers'] = df['answers'].apply(len)
print(df['num_answers'].describe())
```
[79] ✓ 0.0s Python

```
Total Q&A pairs: 394
count    394.000000
mean       2.175127
std        1.670669
min        1.000000
25%        1.000000
50%        1.000000
75%        3.000000
max       11.000000
Name: num_answers, dtype: float64
```

Explanation

count = 394 → Total number of question–answer segments.

mean ≈ 2.17 → On average, each question has about 2.17 answers.

std ≈ 1.67 → The number of answers per question varies; not always just one.

min = 1 → There are questions that only received a single answer.

25%, 50%, 75% → Percentiles (Q1, Median, Q3): most questions were answered 1–3 times.

max = 11 → One particular question received up to 11 answers.

Figure 6. Statistical Summary of Q&A Segment Distribution
Source: Author

D. Chatbot Construction Using Sentence Embeddings

As a practical application of the segmented conversation data, a simple chatbot was developed to respond to user input based on semantic similarity. Unlike basic string matching methods, this approach leverages sentence embeddings to capture the meaning behind user queries. The model chosen for this task is all-MiniLM-L6-v2, a lightweight yet effective transformer model provided by the sentence-transformers library.

The first step in building the chatbot involves extracting all questions from the previously segmented dataset. Speaker labels such as “Human 1:” or “Human 2:” are removed to ensure that the encoding process focuses purely on the content. Each question is then transformed into a vector representation using the embedding model, resulting in a set of question embeddings that serve as the chatbot’s knowledge base.

When a user submits a query, the input sentence is encoded into its corresponding embedding. A cosine similarity comparison is then performed between the user input and the existing question embeddings. The question with the highest similarity score is selected, and the chatbot returns the first associated answer as its response.

This semantic approach allows the chatbot to handle variations in phrasing and vocabulary, enabling it to recognize questions that are not identical but convey similar intent. Ultimately, this demonstrates that regex-based segmentation can serve as a reliable foundation for constructing a lightweight, context-aware question-answering system.

```
src > chatbot.py > ...
from sentence_transformers import SentenceTransformer, util
import pandas as pd

model = SentenceTransformer('all-MiniLM-L6-v2')

df = pd.read_csv("../data/segmented_chat.csv")
import ast
df['answers'] = df['answers'].apply(ast.literal_eval)

questions = df['question'].str.replace(r'^Human \d+: ', '', regex=True)
question_embeddings = model.encode(questions.tolist(), convert_to_tensor=True)

def chatbot_response(user_input):
    user_embedding = model.encode(user_input, convert_to_tensor=True)
    cosine_scores = util.pytorch_cos_sim(user_embedding, question_embeddings)

    best_match_idx = cosine_scores.argmax().item()
    return df.loc[best_match_idx, 'answers'][0]

while True:
    user_input = input("You: ")
    if user_input.lower() in ['exit', 'quit']:
        print("Bot: Goodbye!")
        break
    print("Bot:", chatbot_response(user_input))
```

Figure 6. Semantic Chatbot Implementation
Source: Author

E. Chatbot Evaluation and Testing

To evaluate the chatbot's performance, several test queries were submitted through the command line interface. The responses demonstrate that the chatbot successfully matches user inputs with semantically similar questions from the segmented dataset. For example, when asked "Where do you live?", the bot responded with "I live in Tokyo, Japan," which reflects a relevant and coherent answer drawn directly from the conversation data.

```
(3.13.3) adndax@Dindackermans-MacBook-Pro src % python3 chatbot.py
You: hi, how are you?
Bot: I'm great, thanks. I just ate a delicious breakfast, which always sets the day up right.
You: my name is dinda, you?
Bot: I'm mark. I work in accounting
You: how old are you?
Bot: I am a painter
You: where do you live?
Bot: I live in Tokyo, Japan
You: what is your job?
Bot: Yeah, but I part time work on a farm. Helping with a bit of everything
You: bye!
Bot: pretty good. just went to church and hangout with friends
You: good bye
Bot: same here! I hope the weather will be nice
You: exit
Bot: Goodbye!
(3.13.3) adndax@Dindackermans-MacBook-Pro src %
```

Figure 7. Chatbot Testing
Source: Author

This testing confirms that the chatbot does not rely on keyword matching alone, but instead uses sentence embeddings to understand the meaning behind each query. The answers provided are not randomly generated but are selected from real responses in the dataset that most closely resemble the user's intent. This behavior validates the effectiveness of using semantic similarity over basic string comparison, especially for casual and diverse natural language inputs.

IV. Conclusion

This study investigated the use of regular expressions (regex) to segment chat-based conversations by identifying question-and-answer (Q&A) patterns. In text-based communication, where dialogues often lack clear structure, regex provides a lightweight, interpretable method for isolating meaningful conversational units. By defining patterns that capture common interrogative and responsive phrases, the system is able to automatically transform unstructured dialogues into structured Q&A pairs suitable for further analysis or downstream applications.

The implementation demonstrated that regex can effectively capture Q&A patterns commonly found in informal conversations such as online chats, forums, or messaging platforms. Despite being rule-based, this approach performed well in recognizing key dialogue turns, especially in contexts where the question-and-answer structure follows predictable linguistic patterns. Another advantage is that the method requires no training data or machine learning models, making it simple to implement and computationally efficient.

However, regex-based segmentation also comes with notable limitations. It is sensitive to informal language variations, inconsistent punctuation, and ambiguous sentence boundaries, which can result in missed or misclassified pairs. As such, its effectiveness is highest in semi-structured or relatively clean datasets. For future development, a promising direction would be to combine regex with machine learning or semantic understanding techniques to handle more complex, context-dependent dialogue structures.

Video Link at YouTube

Github: https://github.com/adndax/Chatbot-Regex.git

Youtube:
https://youtu.be/HJgz2Bvtytc?si=yFprioUTzs5Wt_mj

Acknowledgment

As the author of this paper, I would like to express my sincere gratitude to Allah SWT for His blessings, which have enabled me to complete this paper entitled *"Chat Conversation Segmentation Based on Question-and-Answer Patterns Using Regular Expressions"*.

I would also like to extend my appreciation to all parties who have supported and assisted me throughout the writing process, especially to:

1. Dr. Rinaldi Munir, as the lecturer of the course IF2211 Strategi Algoritma, for his guidance, teachings, and the knowledge that has formed a strong foundation for the preparation of this paper.
2. My parents, for their endless prayers and moral support that have encouraged me to complete this paper with dedication.
3. My fellow classmates, especially the SPLIT BILL group, who have provided both technical input amotivation throughout the course.
4. The authors of the journals, articles, and other references that have served as inspiration and contributed meaningful insights for the content of this paper.

I realize that this paper would not have come to fruition without the contributions and support of the aforementioned individuals. Therefore, I sincerely thank everyone for their help, inspiration, and the knowledge they have shared. I hope this paper can provide valuable insights for readers and contribute to the development of knowledge, particularly in the field of technology.

REFERENCES

[1] Massayu. L. Khodra, *String Matching dengan Regular Expression*, Informatika ITB, 2025. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/24-String-Matching-dengan-Regex-(2025).pdf

[2] I. V. Serban, R. Lowe, P. Henderson, L. Charlin, and J. Pineau, "A survey of available corpora for building data-driven dialogue systems," *Dialogue & Discourse*, vol. 9, no. 1, pp. 1–49, 2015.

[3] Y. Chen and H. Wang, "Understanding conversation structures through dialogue act segmentation," *Computational Linguistics*, vol. 45, no. 4, pp. 775–807, 2019.

[4] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 3rd ed. Pearson Education, 2020.

[5] L. Zhou, J. Gao, D. Li, and H. Y. Shum, "The design and implementation of XiaoIce, an empathetic social chatbot," *Computational Linguistics*, vol. 46, no. 1, pp. 53–93, 2020.

STATEMENT OF ORIGINALITY

I hereby declare that the paper I have written is my own original work, not an adaptation or translation of someone else's paper, and is not an act of plagiarism.

Bandung, June 24, 2025

Adinda Putri
13523071