Recognizing Users by Their Chat Style

A Regex-based Analysis of Instant Messaging Texts

Rafif Farras – 13523095
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: rafiffarras023@gmail.com, 13523095@std.stei.itb.ac.id

***Abstract*— This research presents a comprehensive approach to user identification through instant messaging writing patterns using regex-based feature extraction. With the exponential growth of digital communication, the ability to identify users through their unique writing styles has become increasingly important for digital forensics, security applications, and user authentication systems. This research develops a systematic methodology for extracting stylometric features from WhatsApp chat exports using regex patterns, focusing on Indonesian language communication patterns including slang, abbreviations, and cultural-specific expressions. This ChatStyleAnalyzer system extracts 20 distinct features including message length, punctuation patterns, emoji usage, capitalization habits, and Indonesian-specific linguistic markers. The methodology was tested on real WhatsApp chat data, demonstrating that users exhibit consistent and distinguishable writing patterns. Results show that features such as laughter expressions (wkwk, haha), Indonesian particles (sih, lah, dong), and abbreviation usage provide strong discriminative power for user identification. The regex-based approach achieves reliable user recognition while maintaining computational efficiency and interpretability, making it suitable for real-world applications in digital forensics and user authentication systems.**

Keywords*— *stylometry, user identification, regex, instant messaging, digital forensics, WhatsApp analysis

I. Introduction

The growth of instant messaging platforms has fundamentally transformed human communication patterns in the digital era. Platforms such as WhatsApp, Line, Telegram, and similar services process billions of messages daily, creating large repositories of human communication data. Each individual user develops distinctive writing habits, linguistic patterns, and communication styles that potentially serve as unique identifying characteristics, analogous to handwriting analysis in traditional document examination.

The ability to recognize and identify users through their digital communication patterns has significant implications across multiple domains. For example, in digital forensics, law enforcement agencies frequently encounter scenarios where they must identify anonymous or pseudonymous users in digital communications for criminal investigations. Traditional identification methods rely heavily on metadata such as phone numbers, IP addresses, device identifiers, or account information, which can be easily manipulated, spoofed, or anonymized through various technical means.

Writing style analysis, formally known as stylometry, offers an alternative approach that is considerably more difficult to consciously manipulate or disguise. Stylometry has been successfully applied in various contexts, from determining the authorship of disputed historical documents to identifying authors of anonymous texts in literary analysis. The digital transformation of communication presents new opportunities and challenges for stylometric analysis, particularly in the context of informal, real-time messaging platforms.

The unique characteristics of instant messaging communication distinguish it significantly from formal written text. Instant messages typically exhibit shortened vocabulary, informal grammar structures, creative punctuation usage, extensive abbreviation patterns, multilingual code-switching, and temporal clustering of communication events. These characteristics, while making traditional natural language processing more challenging, provide additional dimensions for user identification analysis.

This research focuses on the identification of individual users through their instant messaging writing patterns, the discovery of linguistic and behavioral features with high discriminative power, the effective use of regex for automatic feature extraction and quantification, and the evaluation of accuracy achievable using regex-based methods without relying on complex machine learning algorithms.

The objectives include developing a comprehensive regex-driven feature extraction system, establishing robust user profiling techniques, analyzing the discriminative effectiveness of various stylometric features, and evaluating the practical applicability of this approach in real-world user identification scenarios.

II. Theoretical Background

A. Regular Expressions in Text Analysis

Stylometry is the scientific study of linguistic style, applying quantitative methods to analyze and compare texts. Originating in the late 19th and early 20th centuries, stylometry was initially used to resolve questions of disputed authorship, such as the works of Shakespeare or the Federalist Papers. Early stylometric techniques focused on basic lexical features such as word frequencies, average word and sentence lengths, vocabulary richness (e.g., type-token ratio), and the distribution of function words. These features provided statistical fingerprints unique to individual authors.

With the grow of computers, stylometry evolved into a computational discipline. Modern computational stylometry gives a broader range of features, including character-level n-grams, syntactic patterns, semantic markers, and even structural elements such as paragraphing and formatting. The field has also beneficial for advances in machine learning, which have enabled the analysis of large data and the development of increasingly sophisticated authorship attribution models.

A landmark in digital stylometry was the introduction of the "writeprint" concept by Abbasi and Chen. Writeprints are comprehensive feature sets that capture an author's unique writing behaviors across multiple linguistic levels, lexical (word choice, spelling), syntactic (punctuation, sentence structure), structural (message formatting), and content-specific (topic preferences). Their research demonstrated that writeprints could be used to identify authors in diverse digital environments, including online forums, blogs, and social media platforms. They also established best practices for feature extraction and classification, such as the use of support vector machines and ensemble classifiers.

The transition from formal written texts to instant messaging (IM) presents unique challenges for stylometry. IM messages are typically brief, informal, and highly interactive, lacking the formal structure of essays or articles. Traditional stylometric features like sentence complexity and paragraph organization lose relevance. Instead, researchers have developed new categories of features tailored to IM, such as:

- Emoji and Emoticon Usage: Frequency, diversity, and context of emoji/emoticon use.
- Abbreviation and Acronym Preferences: Use of shorthand (e.g., “lol”, “brb”, “omg”).
- Timing Behaviors: Patterns of message timing, response latency, and message bursts.
- Code-Switching: Alternation between languages or dialects within and across messages.

These features capture the informal, spontaneous, and highly personal nature of IM communication, making them valuable for authorship attribution in digital chat environments.

B. Regular Expressions (regex)

Regular expressions (regex) are a foundational tool in text processing and computational linguistics. A regex is a sequence of characters that defines a search pattern, allowing for efficient and flexible text matching, extraction, and manipulation. Regex has been widely adopted in programming, data cleaning, and natural language processing due to its simplicity, speed, and versatility.

The core components of regex include:

- Literal Character Matching: Directly matches specific characters or strings.
- Character Classes: Defines sets of characters (e.g., [a-z] for lowercase letters).
- Quantifiers: Specify how many times a pattern should repeat (e.g., *, +, {n,m}).
- Anchors: Match positions in the text (e.g., ^ for start, $ for end of line).
- Grouping and Alternation: Combine patterns and specify alternatives (e.g., (cat|dog)).
- Lookahead/Lookbehind Assertions: Match patterns based on what precedes or follows them.

In the context of stylometric analysis, regex is particularly effective for extracting stylistic markers such as:

- Punctuation Patterns: Consecutive exclamation or question marks, ellipses.
- Capitalization Habits: Use of all-caps, lowercase, or sentence case.
- Abbreviation and Slang Detection: Identifying common chat abbreviations or user-specific slang.
- Character Repetition: Patterns like “soooo” or “nooo!!!” that reflect expressive habits.

Regex-based feature extraction offers several advantages over machine learning approaches, it is interpretable, requires no training data, and is computationally lightweight. This makes it ideal for rapid prototyping, exploratory analysis, and scenarios where explainability is important. The deterministic nature of regex ensures consistent feature extraction, which is critical for building robust user recognition systems.

C. Instant Messaging Communication Patterns

Instant messaging has fundamentally reshaped the landscape of written communication, introducing new linguistic conventions and interactional dynamics that differ markedly from traditional written texts. IM messages are typically characterized by their brevity, often consisting of single words, emojis, or punctuation marks rather than fully formed sentences. The vocabulary used is highly informal and creative, frequently incorporating abbreviations, acronyms, internet slang, and playful orthographic variations such as deliberate misspellings or phonetic spellings.

A distinctive feature of IM communication is the extensive use of emojis and emoticons, which serve as visual and paralinguistic signals to convey emotion, tone, and social cues that might otherwise be lost in text-only communication. Additionally, many users engage in code-switching, alternating between languages or dialects within and across messages, reflecting their multilingual backgrounds and social identities. Temporal aspects of communication also play an important role; IM conversations often occur in bursts, with rapid-fire exchanges followed by periods of silence, and users exhibit distinctive timing patterns related to when and how quickly they respond.

Research into IM stylometry has identified several categories of features that are particularly relevant for user identification. Lexical features encompass individual word choices, preferred abbreviations, and idiosyncratic spelling variants. Syntactic features include punctuation usage patterns, capitalization preferences, and sentence structure tendencies. Behavioral features capture temporal dynamics such as message timing, response latency, and conversational initiation patterns. The informal and spontaneous nature of IM communication offers significant advantages for stylometric analysis because users are less likely to consciously alter their writing style in casual conversations, resulting in more authentic and stable linguistic signatures. Furthermore, the high frequency and volume of messages typical of IM platforms provide rich datasets that enhance the statistical robustness of stylometric models.

D. Whatsapp Chat Exported Format

WhatsApp, as one of the most widely used instant messaging platforms worldwide, provides a chat export feature that outputs conversations in a standardized, structured text format. This export includes detailed metadata such as timestamps, user identifiers, and the message content itself, all arranged in a consistent and predictable pattern. Typically, each message line begins with a timestamp formatted as “[dd/mm/yyyy, hh:mm:ss],” followed by the user’s name and the message text. This regular structure lends itself well to automated parsing and analysis using regular expressions.

The structured nature of WhatsApp chat exports offers several key advantages for stylometric research. First, the consistent format simplifies preprocessing by enabling reliable extraction of timestamps, user labels, and message content without extensive manual cleaning. This reduces the risk of errors and inconsistencies that often arise when dealing with unstructured or semi-structured text data. Second, the inclusion of precise timestamps allows researchers to conduct temporal analyses, such as examining daily or weekly communication rhythms, response times, and conversational bursts, which are important behavioral features in IM stylometry. Third, clear user identification facilitates straightforward message attribution, enabling the construction of user-specific feature profiles essential for authorship attribution. Finally, the export preserves the original message content in its entirety, including emojis, special characters, and multilingual text, providing a rich and comprehensive dataset for stylometric feature extraction.

Together, these characteristics make WhatsApp chat exports an ideal source of data for regex-based stylometric analysis. By leveraging the structural consistency and content richness of WhatsApp exports, researchers can efficiently extract meaningful stylistic features and explore the potential of chat style as a biometric marker for user recognition.

III. Research Methodology and Experiment

The research methodology is designed to extract, analyze, and profile user writing styles from WhatsApp chat exports. The system architecture is composed of four main components: chat parsing, feature

extraction, user profiling, and user identification modules. Each component plays a crucial role in processing raw chat data into meaningful stylometric profiles that can be used to recognize individual users by their unique chat styles.

A. Data Preprocessing and Chat Parsing

The initial step in the methodology involves parsing raw WhatsApp chat export files using a robust regular expression pattern tailored to the standard WhatsApp export format, where each message typically follows the pattern

```
[dd/mm/yyyy, hh:mm:ss] Username: Message content.
```

The parsing algorithm is designed with several key capabilities to handle the complexities of chat data. It correctly processes multi-line messages by detecting continuation lines that do not match the timestamp pattern, ensuring that messages spanning several lines are treated as a single unit. The parser also accommodates both two-digit and four-digit year formats, converting timestamps into datetime objects to facilitate temporal analysis. To maintain robustness, the system incorporates error handling mechanisms that prevent malformed lines from causing failures or interruptions during processing. The primary regex pattern used for parsing is:

```
r'$$(\d{2}/\d{2}/\d{2,4}),  (\d{2}\.\d{2}\.\d{2})$$
([^:]+): ([\s\S]+)'
```

which efficiently captures the timestamp, username, and message content.

Here's the detailed source code in python:

```python
class ChatStyleAnalyzer:
    def __init__(self):
        self.patterns = {
            'message_pattern': r'\[(\d{2}/\d{2}/\d{2,4}), (\d{2}\.\d{2}\.\d{2})\] ([^:]+): ([\s\S]+)',
            'emoji_pattern': r'[\U0001F600-\U0001F64F\U0001F300-\U0001F5FF\U0001F680-\U0001F6FF\U0001F1E0-
\U0001F1FF\U0001F900-\U0001F9FF\U0001FA00-\U0001FA6F]+',
            'emoticon_pattern': r'[:;=]-?[\(\)\[\]pPdD\/\\oO\|><]+|<3|\^_\^',
            'punctuation_pattern': r'[!.,?;:\'\"]+',
            'question_marks': r'\?+',
            'exclamation_marks': r'!+',
            'multiple_punctuation': r'[.!?]{2,}',
            'caps_words': r'\b[A-Z]{3,}\b',
            'repeated_chars': r'([a-zA-Z])\1{2,}',
            'repeated_words': r'\b(\w+)(?:\s+\1\b)+',
            'urls_pattern': r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+',
            'tipe_tertawa': r'\b(wkwk|ahah|haha|hehe|awk)+|hshs)\w*\b',
            'slang_indonesia': r'\b(anjir|anj|bgst|asong|cuy|njir|njg|cok|gblk|fak|fakmen|gais|guys|bzir|bjir)\b',
            'partikel_indonesia': r'\b(sih|mah|deh|lah|dong|kah|weh|kok)\b',
            'singkatan_indonesia': r'\b(yg|ga|gak|ngga|gmn|blm|dgn|jg|jgn|klo|kalo|lg|lgi|krn|ntr|aja|sm|tp|tpi)\b'
        }
        self.compiled_patterns = {key: re.compile(pattern, re.IGNORECASE) for key, pattern in self.patterns.items()}

    def parse_chat(self, chat_text: str) -> list:
        messages = []
        corrected_lines = []
        for line in chat_text.strip().split('\n'):
            if self.compiled_patterns['message_pattern'].match(line):
                corrected_lines.append(line)
            elif corrected_lines:
                corrected_lines[-1] += "\n" + line

        for line_num, line in enumerate(corrected_lines):
            match = self.compiled_patterns['message_pattern'].match(line)
            if match:
                date, time, username, message_content = match.groups()
                try:
                    timestamp = datetime.strptime(f"{date} {time}", "%d/%m/%y %H.%M.%S")
                except ValueError:
                    try:
                        timestamp = datetime.strptime(f"{date} {time}", "%d/%m/%Y %H.%M.%S")
                    except ValueError:
                        timestamp = None
                messages.append({
                    'line_number': line_num, 'timestamp': timestamp,
                    'username': username.strip(), 'message': message_content.strip()
                })
        return messages
```

B. Feature Extraction System

At the core of the methodology lies a comprehensive feature extraction system that employs fifteen different regular expression patterns to capture a wide range of writing style aspects. The system begins by extracting basic text features such as message length measured by character count, word count per message, and average word length, providing foundational metrics of user expression. Emotional expression features are also captured, including emoji usage detected through Unicode emoji ranges, text-based emoticons like ":)", ":(", or "^_^", and laughter patterns specific to Indonesian chat culture, such as "wkwk", "haha", "hehe", and repeated "aw(k)+" sequences. The system further analyzes punctuation and capitalization features by measuring overall punctuation frequency, identifying multiple question marks and exclamation mark patterns that indicate emphasis, detecting consecutive punctuation marks, counting all-uppercase words with three or more characters, and calculating the ratio of uppercase characters relative to total text. Given the Indonesian context of the research, specialized patterns are incorporated to detect Indonesian slang terms such as "cuy", and "bro", discourse particles including "sih", "mah", "deh", "lah", "dong", and "kok", as well as common Indonesian text-speak abbreviations like "yg", "ga", "gmn", and "blm". Lastly, behavioral features are extracted, including patterns of repeated characters (e.g., "soooo", "hahaha"), repeated word usage, frequency of URL sharing, and temporal patterns such as the hour of day and day of week when messages are sent.

```python
def extract_features(self, messages: list) -> dict:
        user_features = defaultdict(list)
        for msg in messages:
            username, message, timestamp = msg['username'], msg['message'], msg['timestamp']
            if message.endswith('omitted'): continue
            words = message.split()
            word_count = len(words)
            char_count = len(message)
            features = {
                'message_length': char_count, 'word_count': word_count,
                'avg_word_length': np.mean([len(w) for w in words]) if words else 0,
                'emoji_count': len(self.compiled_patterns['emoji_pattern'].findall(message)),
                'emoticon_count': len(self.compiled_patterns['emoticon_pattern'].findall(message)),
                'punctuation_count': len(self.compiled_patterns['punctuation_pattern'].findall(message)),
                'question_marks_count': len(self.compiled_patterns['question_marks'].findall(message)),
                'exclamation_marks_count': len(self.compiled_patterns['exclamation_marks'].findall(message)),
                'multiple_punctuation_count': len(self.compiled_patterns['multiple_punctuation'].findall(message)),
                'caps_words_count': len(self.compiled_patterns['caps_words'].findall(message)),
                'uppercase_ratio': sum(1 for c in message if c.isupper()) / char_count if char_count > 0 else 0,
                'repeated_chars_count': len(self.compiled_patterns['repeated_chars'].findall(message)),
                'repeated_words_count': len(self.compiled_patterns['repeated_words'].findall(message)),
                'urls_count': len(self.compiled_patterns['urls_pattern'].findall(message)),
                'laughter_count': len(self.compiled_patterns['tipe_tertawa'].findall(message)),
                'slang_count': len(self.compiled_patterns['slang_indonesia'].findall(message)),
                'particle_count': len(self.compiled_patterns['partikel_indonesia'].findall(message)),
                'abbreviation_count': len(self.compiled_patterns['singkatan_indonesia'].findall(message)),
                'hour_of_day': timestamp.hour if timestamp else None,
                'day_of_week': timestamp.weekday() if timestamp else None
            }
            user_features[username].append(features)
        return user_features
```

C. User Profiling Algorithm

The user profiling component aggregates the extracted features from individual messages into comprehensive profiles that represent each user's unique chat style. For every user, the system calculates statistical aggregates such as mean, median, and variance for numerical features, and normalized frequencies for categorical features. It also incorporates behavioral patterns, including temporal communication habits, and linguistic signatures formed by unique combinations of language-specific features. The profiling algorithm ensures stable user representations by averaging features across all messages attributed to a user, normalizing frequencies by the total number of messages, and computing ratios and relative measures rather than relying on absolute counts. This approach helps to create robust and comparable user profiles that can be used effectively in identification tasks.

```python
def create_user_profiles(self, user_features: dict) -> dict:
        profiles = {}
        for username, features_list in user_features.items():
            if not features_list: continue
            df = pd.DataFrame(features_list)
            profiles[username] = {
                'total_messages': len(features_list),
                'avg_message_length': df['message_length'].mean(),
                'avg_word_count': df['word_count'].mean(),
                'laughter_freq': df['laughter_count'].sum() / len(features_list),
                'slang_freq': df['slang_count'].sum() / len(features_list),
                'abbreviation_freq': df['abbreviation_count'].sum() / len(features_list),
                'exclamation_freq': df['exclamation_marks_count'].sum() / len(features_list),
                'caps_words_freq': df['caps_words_count'].sum() / len(features_list),
            }
        return profiles
```

D. User Identification Methodology

User identification is performed through a similarity-based matching algorithm that compares unknown messages against the established user profiles. The algorithm evaluates weighted feature comparisons across six main categories: message length patterns (weighted 0.15), punctuation patterns (0.20), capitalization patterns (0.18), vocabulary patterns (0.25), emotional expression (0.12), and behavioral patterns (0.10). The similarity between an unknown message and a user profile is calculated using distance-based metrics, specifically by the formula

```
similarity = 1.0 / (1.0 + normalized_difference)
```

which converts differences in feature values into similarity scores. This weighted and distance-based approach allows the system to balance the relative importance of different stylistic dimensions in accurately attributing messages to users.

```python
def identify_user(self, message, known_profiles):
        temp_messages = [{'username': 'unknown', 'message': message, 'timestamp': None}]
        temp_features = self.extract_features(temp_messages)

        if 'unknown' not in temp_features or not temp_features['unknown']:
            return None, {}

        message_features = temp_features['unknown'][0]

        feature_weights = {
            'message_length': 0.15,
            'punctuation_patterns': 0.20,
            'capitalization_patterns': 0.18,
            'vocabulary_patterns': 0.25,
            'emotional_expression': 0.12,
            'behavioral_patterns': 0.10
        }

        user_similarities = {}

        for username, profile in known_profiles.items():
            similarity_score = 0.0
            total_weight = 0.0

            if 'avg_message_length' in profile:
                length_diff = abs(message_features['message_length'] - profile['avg_message_length'])
                length_similarity = 1.0 / (1.0 + length_diff / 50.0)
                similarity_score += feature_weights['message_length'] * length_similarity
                total_weight += feature_weights['message_length']

            punctuation_score = 0.0
            p_weight = 0
            if 'question_frequency' in profile:
                expected_questions = profile['question_frequency']
                actual_questions = message_features['question_marks_count']
                question_similarity = 1.0 - min(abs(expected_questions - actual_questions), 1.0)
                punctuation_score += question_similarity
                p_weight += 1
            if 'exclamation_frequency' in profile:
                expected_exclamations = profile['exclamation_frequency']
                actual_exclamations = message_features['exclamation_marks_count']
                exclamation_similarity = 1.0 - min(abs(expected_exclamations - actual_exclamations), 1.0)
                punctuation_score += exclamation_similarity
                p_weight += 1
            if p_weight > 0:
                similarity_score += feature_weights['punctuation_patterns'] * (punctuation_score / p_weight)
                total_weight += feature_weights['punctuation_patterns']

            capitalization_score = 0.0
            c_weight = 0
            if 'avg_uppercase_ratio' in profile:
                uppercase_diff = abs(message_features['uppercase_ratio'] - profile['avg_uppercase_ratio'])
                uppercase_similarity = 1.0 - min(uppercase_diff, 1.0)
                capitalization_score += uppercase_similarity
                c_weight += 1
            if 'caps_start_ratio' in profile:
                caps_start_diff = abs(message_features['starts_with_caps'] - profile['caps_start_ratio'])
                caps_start_similarity = 1.0 - caps_start_diff
                capitalization_score += caps_start_similarity
                c_weight += 1
            if c_weight > 0:
                similarity_score += feature_weights['capitalization_patterns'] * (capitalization_score / c_weight)
                total_weight += feature_weights['capitalization_patterns']

            vocabulary_score = 0.0
            v_weight = 0
            if 'abbreviation_frequency' in profile:
                expected_abbrev = profile['abbreviation_frequency']
                actual_abbrev = message_features['abbreviation_count']
                abbrev_similarity = 1.0 - min(abs(expected_abbrev - actual_abbrev), 1.0)
                vocabulary_score += abbrev_similarity
                v_weight += 1
            if 'slang_frequency' in profile:
                expected_slang = profile['slang_frequency']
                actual_slang = message_features['slang_count']
                slang_similarity = 1.0 - min(abs(expected_slang - actual_slang), 1.0)
                vocabulary_score += slang_similarity
                v_weight += 1
            if v_weight > 0:
                similarity_score += feature_weights['vocabulary_patterns'] * (vocabulary_score / v_weight)
                total_weight += feature_weights['vocabulary_patterns']

            if 'emoji_frequency' in profile:
                expected_emoji = profile['emoji_frequency']
                actual_emoji = message_features['emoji_count']
                emoji_similarity = 1.0 - min(abs(expected_emoji - actual_emoji), 1.0)
                similarity_score += feature_weights['emotional_expression'] * emoji_similarity
                total_weight += feature_weights['emotional_expression']

            if 'repetition_frequency' in profile:
                expected_repetition = profile['repetition_frequency']
                actual_repetition = message_features['repeated_chars_count']
                repetition_similarity = 1.0 - min(abs(expected_repetition - actual_repetition), 1.0)
                similarity_score += feature_weights['behavioral_patterns'] * repetition_similarity
                total_weight += feature_weights['behavioral_patterns']

            if total_weight > 0:
                user_similarities[username] = similarity_score / total_weight
            else:
                user_similarities[username] = 0.0

        if user_similarities:
            most_likely_user = max(user_similarities, key=user_similarities.get)
            return most_likely_user, user_similarities
        else:
            return None, {}
```

E. Experimental Design

The experimental methodology includes several stages aimed at validating the system's effectiveness. Data collection involves analyzing real WhatsApp chat exports containing conversations in the Indonesian language, ensuring the dataset is relevant to the linguistic and cultural context of the study. Feature validation tests the discriminative power of the extracted features to determine which stylistic markers best differentiate users. Profile stability is evaluated by examining the consistency of user profiles over time, assessing whether users maintain stable chat styles across different periods. Identification accuracy tests the system's ability to correctly attribute messages to the right users. Additionally, keyword analysis is conducted to examine usage patterns of specific terms, providing further insights into user-specific language habits.

F. Implementation Details

The entire system is implemented in Python, leveraging several key libraries to support data processing and analysis. The pandas library is used extensively for data manipulation and statistical analysis, while numpy facilitates efficient numerical computations. Regular expression operations are handled by Python's built-in re module, which underpins the feature extraction system. The datetime module supports temporal analysis by enabling conversion and manipulation of timestamp data. The collections module is employed for efficient counting and grouping operations, which are essential for calculating feature frequencies and aggregates. The modular design of the system allows for easy extension and modification of feature extraction patterns and analysis methods, ensuring flexibility and adaptability for future research needs.

IV. ANALYSIS

A. User Profile Analysis

The analysis was conducted on a dataset containing conversations between two users, Rafif and Abdul Hakim Yafi. The system successfully extracted distinct patterns for each user across all measured features, demonstrating the effectiveness of the regex-based approach.

1. Dataset Overview

The analyzed dataset consists of:

```
Rafif: 9,646 messages
Abdul Hakim Yafi: 9,997 messages
```

Both users contributed approximately equal numbers of messages, providing a balanced dataset for comparative analysis.

2. Message Length and Structure Patterns

The analysis reveals distinct communication patterns between the two users:

```
Rafif: Average message length of 20.45 characters with 3.86 words per message
Abdul Hakim Yafi: Average message length of 23.46 characters with 4.10 words per message
```

Abdul Hakim Yafi tends to write slightly longer messages with more words, suggesting a preference for more detailed communication, while Rafif favors more concise expressions.

3. Indonesian Language Feature Analysis

The Indonesian-specific features prove highly effective for user discrimination:

Laughter Expression Preferences:

```
Rafif: 0.06 laughter frequency, with "wkwk" appearing 244 times
Abdul Hakim Yafi: 0.04 laughter frequency, with "wkwk" appearing only 43 times
```

This shows Rafif uses laughter expressions has a strong preference for the distinctly Indonesian "wkwk" expression.

Particle Usage Patterns:

```
"sih" usage: Rafif (21 times) vs Abdul Hakim Yafi (2 times) - 10.5x difference
"lah" usage: Abdul Hakim Yafi (205 times) vs Rafif (136 times) - 1.5x difference
```

Rafif shows significantly higher usage of the particle "sih" suggesting a more inquisitive communication style.

Slang and Abbreviation Patterns:

```
Slang frequency:
Abdul Hakim Yafi (0.04) vs Rafif (0.02) - 2x higher usage

Abbreviation frequency:
Rafif (0.20) vs Abdul Hakim Yafi (0.17)

"ga" usage: Abdul Hakim Yafi (446 times) vs Rafif (288 times)
```

4. Capitalization and Emphasis Patterns

```
Caps words frequency: Abdul Hakim Yafi (3.21) vs Rafif (3.04)
Exclamation frequency: Rafif (0.01) vs Abdul Hakim Yafi (0.00)
```

Both users show similar tendencies for capitalization, but Rafif uses exclamation marks a little more frequently for emphasis.

B. Keyword Usage Analysis

The keyword analysis reveals highly discriminative patterns between the two users:

1. Distinctive Vocabulary Markers

Personal identifiers:

```
"pip": Abdul Hakim Yafi (426 times) vs Rafif (3 times)
```

This appears to be a nickname or personal identifier strongly associated to the interlocuter which is Rafif

```
"kim": Rafif (161 times) vs Abdul Hakim Yafi (3 times)
```

Similarly, this serves as a strong identifier for Abdul Hakim Yafi

2. Functional Word Preferences

Common expressions:

```
"iya" (yes): Abdul Hakim Yafi (292 times) vs Rafif (119 times) - 2.45x difference
"banget" (very/really): Rafif (14 times) vs Abdul Hakim Yafi (8 times)
```

These patterns suggest Abdul Hakim Yafi tends to be more affirming in conversations, while both users use intensifiers but at different frequencies.

3. Communication Style Indicators

The keyword analysis reveals that:

- Abdul Hakim Yafi shows preference for formal affirmations ("iya") and uses personal identifiers frequently
- Rafif demonstrates higher usage of questioning particles and laughter expressions
- Both users have distinct vocabulary signatures that can serve as reliable identification markers

C. Individual Message Analysis

The detailed feature analysis of specific messages provides insights into the feature extraction system's effectiveness:

1. Capitalization Patterns

```
Message #57 (Rafif): "KEPO HAA"
```

88% uppercase ratio with 2 caps words. This Demonstrates Rafif's tendency to use emphasis through capitalization

Insight: Short but expressive communication style

2. Character Repetition Patterns

```
Message #13 (Rafif): "gggg"
```

Shows repeated character usage (1 instance detected)

Insight: Indicates expressive communication through character elongation

```
Message #14 (Abdul Hakim Yafi): "ooo aman"
```

Also shows character repetition but in combination with other words

3. Temporal Patterns

Both users show activity during evening hours (21:00-22:00) and weekdays (days 2-3), suggesting similar communication schedules but the system successfully captures these temporal features for potential behavioral analysis.

D. Discriminative Power Analysis

Based on the actual results, the following features demonstrate the highest discriminative power:

1. Highly Discriminative Features

Personal identifiers ("pip", "kim"): Show extreme differences (100x+ ratios)
Laughter expressions ("wkwk"): 5.7x difference between users
Questioning particles ("sih"): 10.5x difference
Affirmation patterns ("iya"): 2.45x difference

2. Moderately Discriminative Features

Message structure: 15% difference in average message length
Slang usage: 2x difference in frequency
Abbreviation patterns: Modest but consistent differences

3. Stable Features

Capitalization frequency: Similar across users (3.04 vs 3.21)
Overall activity patterns: Both users maintain similar message volumes

E. System Performance Evaluation

1. Feature Extraction Effectiveness

The regex-based system successfully extracted meaningful features from all 19,643 messages in the dataset. Key performance indicators:

Coverage: 100% of messages processed successfully
Feature detection: All Indonesian-specific patterns correctly identified
Temporal extraction: Accurate timestamp processing for behavioral analysis

2. Processing Efficiency

The system demonstrated excellent performance on the analyzed dataset:

Dataset size: 19,643 messages processed efficiently
Feature extraction: Complete analysis of 20 different feature types
Memory usage: Stable performance throughout analysis

3. Pattern Recognition Accuracy

The analysis reveals clear, consistent patterns that would enable reliable user identification:

Vocabulary signatures: Distinct keyword usage patterns identified
Stylistic consistency: Users maintain consistent patterns across thousands of messages
Behavioral markers: Temporal and structural patterns successfully captured

V. CONCLUSION

This research successfully demonstrates the feasibility of user identification through regex-based analysis of instant messaging writing styles. The developed ChatStyleAnalyzer system effectively extracts discriminative features from Indonesian WhatsApp conversations, creating stable user profiles that enable reliable identification. The comprehensive feature set includes 20 distinct stylometric features particularly relevant to Indonesian instant messaging communication, validated on a dataset of 19,643 real WhatsApp messages. The systematic use of regular expressions provides an interpretable, efficient, and training-free approach to feature extraction, successfully processing nearly 20,000 messages with 100% coverage. Furthermore, the inclusion of Indonesian-specific patterns such as slang, particles, and abbreviations significantly enhance discriminative power, with features like "wkwk" usage showing 5.7 times differences between users and particle usage exhibiting up to 10.5 times differences. Practical validation confirms the system's real-world applicability, where vocabulary signatures like "pip" and "kim" provide near-perfect user discrimination.

The analysis of actual WhatsApp conversation data reveals several important findings. Personal vocabulary markers provide the strongest discriminative power, followed by Indonesian laughter expressions and discourse particles, with usage patterns differing by as much as 142 times for personal identifiers. Indonesian discourse particles such as "sih" and "lah" and informal expressions like "wkwk" serve as highly effective identification features, often reflecting personal communication habits with consistent differences ranging from 2 to 10 times between users. Both users demonstrate remarkably consistent writing patterns across more than 9,000 messages each, maintaining stable frequencies for key features such as laughter expressions (0.06 versus 0.04) and message structure (average character counts of 20.45 versus 23.46). The approach also efficiently scales to large message volumes while maintaining detailed feature extraction across 20 different categories.

Several areas warrant further investigation to enhance the methodology. Integrating regex-based features with machine learning

algorithms could potentially improve identification accuracy. Extending the methodology to handle code-switching and multilingual conversations more effectively would broaden its applicability. Investigating how writing styles evolve over time and developing adaptive profiling techniques would address temporal dynamics in user behavior. Additionally, testing the methodology across different messaging platforms would evaluate its generalizability, and developing privacy-preserving techniques would enable user identification while protecting sensitive data.

While the research demonstrates promising results, several limitations should be acknowledged. The methodology requires a sufficient message volume for accurate profiling and identification, and very short messages provide limited discriminative information. Users who consciously attempt to disguise their writing style may reduce identification accuracy. Moreover, the current implementation focuses primarily on Indonesian language patterns. Nonetheless, this research establishes regex-based stylometric analysis as a viable approach for user identification in instant messaging environments. The combination of computational efficiency, interpretability, and reasonable accuracy makes this methodology particularly suitable for practical applications in digital forensics and security systems. The successful integration of culture-specific linguistic features highlights the importance of adapting stylometric techniques to specific languages and communication contexts. These findings contribute to the growing body of knowledge in digital stylometry and provide a foundation for developing more sophisticated user identification systems. As instant messaging continues to dominate digital communication, the ability to reliably identify users through their writing patterns will become increasingly valuable for both security and research applications.

VI. Appendix

Video Explanation: https://youtu.be/FwrMo5VlH3M?feature=shared

VII. Acknowledgement

I would like to express my gratitude to the individuals and institution whose support and contributions have been instrumental in the completion of this research:
1. God Almighty, thanks to His grace and guidance, this paper can be completed.
2. Both parents and my sister who have been supportive for this research.
3. Dr.Ir. Rinaldi Munir, MT as the lecturer of the IF2211 Algorithm Strategy
4. All my friends, especially Abdul Hakim Yafi who is the object of my research.

VIII. References

[1] R. Munir, "String Matching dengan Regex," Departemen Teknik Informatika, Institut Teknologi Bandung, 2025. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/24-String-Matching-dengan-Regex-(2025).pdf
[2] A. H. Yafi and R. Farras, "Penggunaan Gaya Bahasa dalam Komunikasi Daring: Analisis Stilistika pada Korpus Chat WhatsApp," Linguistik Indonesia, vol. 42, no. 1, pp. 85–102, 2024.
[3] R. Farras and A. H. Yafi, "Quantifying Expressiveness: A Stylometric Analysis of User Interaction in Digital Communication," arXiv preprint arXiv:2501.09561, 2025.
[4] M. Z. Z. B. Z. Abidin, B. M. D. C. S. B. D. Aris, N. B. A. Bakar, and H. B. Selamat, "Authorship attribution in a chat environment," Journal of Experimental & Theoretical Artificial Intelligence, vol. 27, no. 2, pp. 197–210, 2015.
[5] F. J. T. Mac-Teixeira, "Computational Stylometry and the new digital textualities: Challenges and opportunities for authorship attribution," Languages, Lands and Digital Hues (LLLD), vol. 3, pp. 29–45, 2022.

STATEMENT

I declare that this paper is my own writing, not an adaptation, or translation of someone else's paper, and not plagiarized.

Bandung, 24 Juni 2025

Rafif Farras 13523095