BAB 2
KAJIAN PUSTAKA

Bab dua menjelaskan mengenai kajian pustaka dan studi literatur yang dilakukan oleh penulis. Beberapa teori dan rumus logika algoritma telah diambil dari berbagai ulasan literatur yang dilakukan, yang kemudian diterapkan dalam pengembangan program untuk laporan ini.

2.1 Kata Terikat

Kata terikat mengacu pada kata atau morfem yang tidak dapat berfungsi secara mandiri sebagai kata bebas dan membutuhkan kata lain untuk membentuk arti yang utuh dalam kalimat. Kata-kata ini umumnya berfungsi sebagai afiks yang terikat pada kata dasar. Kata terikat memiliki beragam pola afiks, termasuk antara-, maha-, hiper-, hipo-, super-, dan lain-lain, yang berperan dalam memperkaya makna kata dasar yang diikuti [11].

2.2 *Natural Language Processing*

Natural Language Processing (NLP) merupakan bidang yang terus berkembang di dalam kecerdasan buatan dan linguistik komputasional. Tujuan NLP adalah untuk memungkinkan komputer memahami, menginterpretasikan, dan mengolah bahasa manusia. Kemajuan NLP dipengaruhi oleh berbagai faktor, termasuk kemajuan dalam pembelajaran mesin, ketersediaan data besar, dan peningkatan kebutuhan analisis teks otomatis. NLP telah diaplikasikan dalam berbagai area, termasuk layanan pelanggan, analisis media sosial, dan perangkat lunak edukasi. Penggunaan *chatbot* dan asisten virtual adalah contoh aplikasi NLP yang telah mengubah interaksi manusia dengan teknologi [12].

2.3 *Pattern Matching*

Algoritma pencocokan pola berfokus untuk menemukan *substring* atau pola dalam teks yang lebih besar, esensial dalam berbagai aplikasi komputasi. Studi tentang algoritma pencocokan pola, seperti *Knuth-Morris-Pratt (KMP), Boyer-Moore, dan Rabin-Karp*, telah menjadi topik penting dalam ilmu komputer. Algoritma-algoritma ini memiliki keunggulan tersendiri dalam efisiensi waktu

dan kompleksitas ruang, tergantung pada konteks aplikasi mereka [7]. Di bidang teknologi informasi, algoritma pencocokan pola telah digunakan dalam berbagai aplikasi, mulai dari keamanan jaringan hingga analisis DNA. Khususnya, penggunaannya dalam sistem keamanan *cyber* untuk mendeteksi ancaman telah mendapat perhatian signifikan [13].

2.4 *Rabin-Karp*

Algoritma *Rabin-Karp*, yang dikembangkan oleh Michael Rabin dan Richard Karp, menggunakan teknik hashing untuk mencari *substring* dalam string induk, menjadikannya alat penting dalam berbagai aplikasi komputasi. Algoritma ini menggunakan '*rolling hash*' untuk menghitung nilai hash *substring* secara berkelanjutan. Pendekatan ini mengurangi kebutuhan untuk menghitung hash dari awal di setiap langkah, dengan menghilangkan kontribusi karakter awal dan menambahkan karakter berikutnya dalam urutan. Jika ada kesamaan nilai hash, algoritma melakukan pemeriksaan tambahan untuk memastikan kesesuaian sebenarnya antara *substring* yang dicari dengan *substring* dalam teks. Langkah ini penting untuk mencegah kesalahan identifikasi karena *collision hash*, di mana string yang berbeda bisa menghasilkan nilai hash yang sama [14].

Kemampuan algoritma ini dalam melakukan pencarian multipola dalam satu operasi menjadikannya efisien untuk aplikasi yang memerlukan identifikasi berbagai pola secara simultan. Algoritma *Rabin-Karp* menawarkan kinerja yang lebih baik dalam hal waktu eksekusi, terutama untuk teks panjang, dan dalam kasus pencarian *substring* yang jarang terjadi. Berikut adalah rumus yang digunakan untuk menghitung hash dalam algoritma *Rabin-Karp* [14][15].

$$hash(s[i..j]) = (s[i] \times b^{m-1} + s[i+1] \times b^{m-2} + .. + s[j]$$

times $b^0 mod\ qDimana:$

- s[i..j] adalah *substring* dari indeks.
- s[k] adalah nilai ASCII dari karakter ke-.
- k pada *string* (dengan asumsi *string* diindeks mulai dari 0).
- b adalah konstanta yang sering dipilih sebagai bilangan prima, mewakili basis sistem bilangan yang digunakan untuk perhitungan hash.

- m adalah panjang pola yang dicari.
- q adalah bilangan prima besar yang digunakan sebagai modulus untuk mencegah *overflow* bilangan bulat besar dan menjaga nilai hash dalam rentang yang dapat dikelola.

Di dunia komputasi, algoritma *Rabin-Karp* telah diterapkan dalam berbagai bidang, termasuk pencarian teks, bioinformatika, dan deteksi plagiarisme [16]. Efisiensinya dalam mengidentifikasi pola membuatnya menjadi pilihan populer untuk aplikasi yang memerlukan pemrosesan *string* cepat dan akurat. Meskipun efektif, algoritma *Rabin-Karp* menghadapi tantangan dalam situasi tertentu, seperti saat berhadapan dengan dataset besar atau karakteristik hash yang serupa.