BAB II
TINJAUAN PUSTAKA

Maertens, R., dkk. telah membuat sebuah alat deteksi plagiarisme kode sumber bernama *Dolos*. Alat sumber terbuka (*open-source*) ini dapat secara fleksibel digunakan untuk hampir semua bahasa pemrograman. Dalam eksekusinya, alat ini melakukan pemeriksaan plagiasi melalui tiga (3) langkah: *tokenization*, *fingerprinting*, dan *indexing*. Alat ini dapat membandingkan satu berkas dan berkas yang lain, serta antara satu *project* dengan *project* yang lain selama berkas (*file*) kode sumber masih berbahasa yang sama. Alat ini oleh penulisnya mampu mengungguli alat pendeteksi plagiarisme lain yang ada saat ini [9][14].

Eppa, A., dkk. telah membuat sebuah alat pendeteksi plagiarisme menggunakan pembelajaran mesin (*machine learning*) dan pembelajaran mendalam (*deep learning*). Penelitian ini dilaksanakan untuk menguji beberapa konsep dalam pembelajaran mesin dan pembelajaran mendalam, antara lain *KNN*, *SVM*, *RNN*, dan beberapa konsep lain dalam mendeteksi plagiarisme pada bahasa pemrograman C. Alat yang dikembangkan diklaim oleh penulisnya lebih akurat dalam mendeteksi plagiarisme dibandingkan pendekatan lain berbasis teks [15].

Rizvee, R., dkk. telah membuat sebuah alat identifikasi plagiarisme kode sumber secara otomatis yang mampu diimplementasi dengan mudah dan efisien. Alat ini menggunakan pendekatan berbasis teks dengan metode pencocokan *string* (*string-matching*). Alat ini juga menggunakan mekanisme *weighted score* untuk menunjukkan laporan plagiarisme. Pada akhir penelitian, alat ini diklaim oleh penulisnya memiliki tingkat presisi yang tinggi (82%), *recall score* yang tinggi (76%), serta hemat memori dan waktu [16].

Lisan Sulistiani dan Oscar Karnalim telah membuat sebuah alat deteksi plagiarisme kode sumber bernama *ES-Plag*. Alat ini bertujuan untuk mendeteksi plagiarisme kode sumber pada lingkungan akademis menggunakan algoritma *Running-Karp-Rabin Greedy-String-Tiling* yang telah dioptimisasi menggunakan

cosine-based filtering. Selain memeriksa tingkat kemiripan kode sumber, alat ini juga memiliki sebuah antarmuka yang menampilkan *similarity matrix* serta perbandingan kode sumber. Alat yang dibangun dapat memeriksa proyek perangkat lunak multi-*file* dengan cara menggabungkan semua kode tersebut ke dalam satu *file*. Alat yang dibangun diklaim oleh penulisnya lebih cepat dibanding pendekatan *Running-Karp-Rabin Greedy-String-Tiling* yang ada sekarang, walau dengan penurunan kecil pada akurasi hasil pemeriksaan [17].

Zhao, J., dkk. telah membuat sebuah alat pendeteksi plagiarisme menggunakan metode *abstract syntax tree* (*AST*). Alat ini bekerja dalam empat (4) tahap, yakni 1) membuat sebuah pohon abstrak dan memberi nilai *hash* pada setiap simpul yang terbentuk, dan 2) klasifikasi setiap *sub-tree* berdasarkan nomor simpul, 3) menggunakan perbandingan *hash*, dan 4) menyimpan *sub-tree* dengan nilai *hash* yang sama. Alat ini diklaim oleh penulisnya dapat secara efektif digunakan untuk mendeteksi plagiarisme, serta mendeteksi kemungkinan kesalahan *syntax* dalam penulisan operator aritmetika [18].

Tito D. K. Siregar telah mengembangkan sebuah algoritma pendeteksi plagiarisme melalui pengukuran jarak *levenshtein* menurut implementasi algoritma *Wagner-Fischer*. Dalam implementasinya, alat ini bekerja melalui dua (2) tahap, yakni *preprocessing* untuk membuang bagian kode yang tidak dibutuhkan (misalnya *whitespace* dan *line break*), kemudian mengukur jarak *levenshtein* untuk mendapatkan tingkat kemiripan. Berdasarkan hasil pengujian yang juga dipublikasi, alat yang dikembangkan sudah memenuhi ekspektasi pada setiap kasus uji yang diberikan, sehingga dapat disimpulkan bahwa mengukur jarak *levenshtein* sudah cukup bagus untuk mendeteksi plagiarisme [19].

Pada tahun 1997, sebuah alat pendeteksi plagiarisme yang masih dipakai hingga saat ini telah dibuat di Stanford University, Amerika Serikat, bernama *MOSS* (*Measurement of Software Similarity*). Alat ini mendeteksi plagiarisme dengan membangun algoritma *fingerprinting* bernama *winnowing*. Algoritma ini merupakan algoritma yang masih lazim dipakai pada beberapa alat pendeteksi plagiarisme yang dikembangkan hingga saat ini, salah satunya *Dolos* [9]. Alat yang

dibangun bisa menerima masukan direktori (untuk memeriksa perangkat lunak multi-*file*), namun terbatas pada satu bahasa pemrograman. Layanan pemeriksaan dapat diakses oleh publik melalui sebuah situs web, dan diklaim oleh penulisnya dipastikan mendeteksi plagiarisme kode sumber [20].

Pada tahun 1996, sebuah alat pendeteksi plagiarisme yang masih rutin dikembangkan hingga saat ini dibuat di *Karlsruhe Institute of Technology*, Jerman, bernama *JPlag*. Alat ini bekerja dengan menggunakan algoritma *Greedy String Tiling*, yang masih digunakan beberapa alat pendeteksi plagiarisme hingga saat ini, salah satunya *ES-Plag* [17]. Alat ini tersedia secara terbuka (*open source*) di *GitHub* [21] serta dapat diunduh oleh pengguna untuk melakukan pemeriksaan pada lingkungan lokal. Alat ini dapat memeriksa proyek perangkat lunak (terbatas pada satu bahasa pemrograman saja), dan diklaim oleh penulisnya bahwa *JPlag* "sangat sulit untuk ditipu", dengan akurasi 90% pada 77 *dataset* yang diuji [22].

Berdasarkan berbagai kajian pustaka yang telah dipaparkan, terdapat banyak pendekatan yang dapat diambil untuk melakukan pemeriksaan proyek perangkat lunak. Namun, berbagai penelitian yang telah dipaparkan hanya mampu memeriksa tingkat kemiripan kode sumber antara satu *file* dengan *file* yang lain, hingga sebatas antar-*project* namun dalam satu bahasa pengkodean. *ProjectGuard* adalah nama dari alat yang dikembangkan pada penelitian ini, dan mampu untuk memeriksa tingkat kemiripan pada proyek perangkat lunak multi-*file* dengan bahasa pengkodean yang berbeda-beda melalui adaptasi alat *Dolos* serta menggunakan jarak *levenshtein*. Diharapkan melalui alat ini, para akademisi dapat lebih mudah mengidentifikasi plagiarisme kode sumber menggunakan visualisasi modern.

Berdasarkan berbagai penelitian tersebut, dapat dibuat sebuah tabel perbandingan seperti pada Tabel 2.1.

Tabel 2.1 Perbandingan penelitian terdahulu

Nama	***Dolos***	***(Eppa dkk, 2022)***	***(Rizvee dkk., 2022)***	**ES-Plag**	**AST-CC**	***(Tito D. K. Siregar, 2014)***	**Moss**	**JPlag**	**Project Guard**
Referensi	[9][14]	[15]	[16]	[17]	[18]	[19]	[20]	[22]	Penulis
Tahun awal rilis	2022	2022	2022	2018	2015	2014	1997	1996	2024
Metode	*Tokenization*, *fingerprinting*, dan *indexing*	*Machine learning* dan *deep learning*	*String-matching* dan *weighted score*	*Running-Karp-Rabin Greedy-String-Tiling*	*Abstract syntax tree*	*Levenshtein distance*	Algoritma *finger-printing* bernama *winnowing*	*Greedy string tiling*	*Tokenization*, *fingerprinting*, *indexing*, serta *levenshtein distance*
Jumlah bahasa yang dapat diperiksa[1]	41	1 (C)	Tidak disebutkan	3 (C++, Java, Python)	1 (Java)	1 (C++)	25	18	10 (7 jenis *project*)
Memiliki *project-based input*[2]	Ya	Tidak	Tidak	Ya	Tidak	Tidak	Ya	Ya	Ya

[1] Hanya menghitung bahasa yang memiliki *parser* atau terdapat *preprocessing*.

[2] Apakah alat tersebut dapat menerima masukan proyek perangkat lunak, bukan hanya satu *file*.

Nama	***Dolos***	***(Eppa dkk, 2022)***	***(Rizvee dkk., 2022)***	**ES-Plag**	**AST-CC**	***(Tito D. K. Siregar, 2014)***	**Moss**	**JPlag**	**Project Guard**
Dukungan pemeriksaan multi-bahasa dalam 1 proyek	Tidak	*N/A*	*N/A*	Tidak	*N/A*	*N/A*	Tidak	Tidak	Ya
Mudah dikustomisasi[3]	Ya	Tidak	Tidak	Tidak	Tidak	Tidak	Ya	Ya	Ya
Tampilan untuk melakukan pemeriksaan	Web, CLI	Aplikasi	Tidak ada	Aplikasi Windows	Tidak ada	Tidak ada	CLI	CLI	Web
Tampilan untuk menampilkan laporan	Web	Aplikasi	Tidak ada	Aplikasi Windows	Tidak ada	Tidak ada	Web	Web	Web
Klasterisasi hasil[4]	Ada	Tidak ada	Tidak ada	Tidak ada	Tidak ada	Tidak ada	Tidak ada	Ada	Ada

[3] Apakah alat tersebut memungkinkan pengguna menambahkan atau mengubah pengaturan di atas pengaturan *default*.

[4] Apakah alat tersebut memilik fitur pengelompokan hasil perbandingan (menggunakan pendekatan *machine learning* maupun tidak).

Nama	***Dolos***	***(Eppa dkk, 2022)***	***(Rizvee dkk., 2022)***	**ES-Plag**	**AST-CC**	***(Tito D. K. Siregar, 2014)***	**Moss**	**JPlag**	**Project Guard**
Bentuk visualisasi data	Histogram, graf hubungan klaster	Tidak disebutkan	Tidak ada	*Similarity matrix*	Tidak ada	Tidak ada	Tidak ada	Histogram, graf hubungan klaster	Histogram, graf hubungan klaster, *similarity matrix*
Fitur Laporan	- Ringkasan - Berdasarkan klaster - Berdasarkan pasangan - *Pair-wise inspection*	Tidak disebutkan	Tidak ada	- Ringkasan - *Pair-wise inspection* - *Matched token inspection*	Tidak ada	Tidak ada	- Ringkasan - *Pair-wise inspection*	- Ringkasan - Berdasarkan klaster - *Pair-wise inspection*	- Ringkasan - Berdasarkan klaster - Berdasrkan *project* - Berdasarkan pasangan - *Pair-wise inspection*