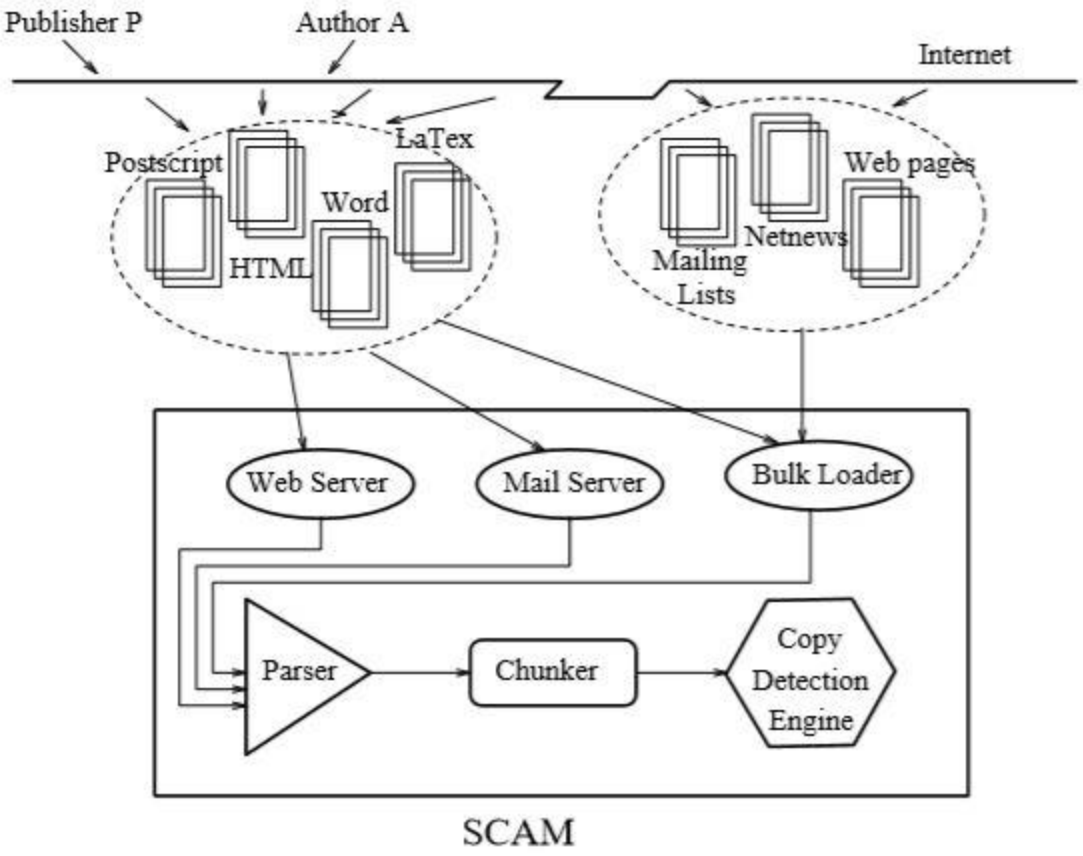

Figure 1: Functionality of SCAM.

Architecture of SCAM

Pada bagian ini kami pertama-tama menguraikan fungsionalitas server registrasi SCAM dari perspektif pengguna; selanjutnya kami uraikan implementasinya. Gambar 1 menunjukkan keseluruhan fungsionalitas SCAM. SCAM bulkloads dokumen tersedia di sumber-sumber publik (seperti Netnews, halaman Web dan pesan pada milis) setiap hari ke dalam repositori publiknya. Penulis dan penerbit kemudian dapat berinteraksi dengan SCAM melalui Server Web-nya (dengan mengisi formulir) atau server Mail-nya (dengan mengirim email). Melalui ujung depan ini, pengguna jarak jauh dapat membuat repositori pribadi dari dokumen terdaftar, atau membandingkan dokumen dengan repositori pribadi atau publik SCAM. SCAM menerima dokumen (untuk pemuatan massal, pendaftaran pribadi, atau perbandingan) dalam

beberapa format (saat ini HTML, ASCII biasa, nota bene(postscript)). Semua dokumen diurai menjadi representasi umum (teks ASCII saat ini), dan dialirkan melalui chunker (perincian di bagian selanjutnya) sebelum potongan dikirim ke Mesin Deteksi Salin untuk perbandingan atau pendaftaran. SCAM melaporkan documento verlaps dalam twow ays: (1) pengguna diberitahukan (pesan email atau formulir Web) jika tumpang tindih terdeteksi antara dokumen kueri dan beberapa dokumen dalam repositori target: (2) pengguna diberitahu jika beberapa dokumen yang mereka miliki terdaftar di repositori pribadi mereka tumpang tindih dengan beberapa dokumen yang kemudian terdaftar di repositori publik. Setelah seorang pengguna diberitahu, ia kemudian dapat mencari dokumen yang dimaksud untuk memeriksa apakah memang ada masalah hukum atau etika. Gambar 2 menggambarkan struktur data utama yang digunakan oleh Mesin Deteksi Salin. Perlu diingat bahwa tujuan kami adalah memiliki struktur yang dapat menskalakan sejumlah besar dokumen. Untuk kenyamanan pembaca, kami telah menomori komponen yang ditunjukkan pada gambar, dan menggunakan angka-angka ini sebagai superskrip dalam teks berikut ini. SCAM menggunakan Buffer Manager (BM) yang berinteraksi dengan sistem file UNIX melalui file paged (blok data). BM cache struktur data persisten berikut (yang berada di disk) ke dalam buffer pool (memori utama) untuk akses cepat:

- **Inverted Index on chunks** : Dalam struktur ini (umumnya digunakan dalam sistem IR [12,13]), kami mempertahankan posting yang menunjukkan frekuensi kemunculan (bukan nol) potongan dalam dokumen. Misalnya, sebuah posting mungkin berbentuk ("gambar", 11, 5) untuk menunjukkan bahwa potongan "gambar" muncul 5 kali dalam dokumen dengan pengidentifikasi unik 11. Indeks terbalik terdiri dari struktur hashing yang dapat diperluas yang diindeks pada potongan untuk pencarian yang efisien. Setiap potongan menunjuk ke satu set halaman yang menyimpan postingannya. Semua postingan untuk chunk dikelompokkan bersama (di halaman disk yang sama) untuk akses yang efisien, dengan halaman overflow ditambahkan bila perlu.

- **Name Mapper** : Dalam struktur ini, kami menyimpan pemetaan dari id dokumen unik yang digunakan dalam posting ke nama file aktual (disimpan secara lokal) atau URL halaman Web. Misalnya, entri dapat berupa http: / www.db.stanford.edu) yang

menunjukkan bahwa dokumen dengan id adalah halaman Web yang terletak di URL yang diberikan. Struktur ini digunakan agar postingan dalam indeks terbalik bisa lebih kecil dengan menyimpan id integer daripada string byte. Kami menerapkan nama mapper sebagai struktur hashing extensible yang diindeks pada id dokumen.

- **Relevansi Array** : Tumpang tindih antara dokumen permintaan dan satu set dokumen yang terdaftar biasanya disimpan dengan bertambahnya beberapa "skor" (dibahas dalam Bagian 3) setiap kali potongan umum ditemukan. Sebagai contoh, katakanlah dokumen permintaan Q memiliki 5 potongan yang sama dengan R1 dan 7 potongan yang sama dengan R2, di mana R1 dan R2 adalah beberapa dokumen terdaftar. Ketika potongan baru ditemukan sama dengan R2, skor R2 bertambah menjadi 8. Biasanya struktur seperti itu yang membutuhkan akses acak ke item data individual (seperti R2) diimplementasikan sebagai array dalam memori utama. Namun karena kumpulan dokumen yang terdaftar bisa besar, mungkin tidak mungkin untuk mengalokasikan array pada saat run time, Kami menerapkan array relevansi sebagai struktur hashing yang diperluas yang diindeks pada id dokumen untuk menyediakan akses cepat ke skor relevansi. Karena struktur ini juga diimplementasikan sebagai file halaman, ketika struktur tumbuh, porsi dapat halaman ke disk.Kami sekarang menguraikan bagaimana struktur di atas digunakan selama registrasi dan fase penyelidikan

- **Registration**: Ketika dokumen baru akan didaftarkan, sebuah entri dibuat di Name Mapper dengan nama file dokumen atau URL (jika itu adalah halaman Web). Integer unik dihasilkan yang memetakan ke nama file. Aliran potongan kemudian dimasukkan ke dalam indeks terbalik berdasarkan memori utama yang melacak id dokumen dan jumlah frekuensi potongan dalam dokumen itu. Indeks terbalik ini secara berkala digabungkan dengan indeks terbalik persisten. Ini dengan I / O batch, mengurangi jumlah I / O acak. Untuk menghemat ruang, kami biasanya membersihkan dokumen yang lebih lama dari repositori publik, karena mereka tidak terlalu menarik.

- **Probing:** Ketika dokumen permintaan diperiksa terhadap repositori, potongan-potongannya disimpan ke dalam indeks memori terbalik utama. Untuk setiap potongan yang disimpan dalam indeks terbalik, kami mengambil postingan dalam indeks persisten dalam persisten untuk menemukan kumpulan dokumen yang berisi potongan itu, dan secara bertahap menghitung tingkat tumpang tindih antara dokumen kueri dan kumpulan dokumen terdaftar menggunakan Relevance Array . Pengguna kemudian diberitahu tentang dokumen-dokumen yang tumpang tindih dengan dokumen permintaan tentang beberapa ambang batas pengguna tertentu.

Algoritma

Biarkan R menjadi dokumen permintaan dan S menjadi dokumen asli, penulis SCAM pertama-tama menentukan kedekatan set c (R, S) untuk memuat kata-kata wi yang memiliki jumlah kemunculan yang serupa di kedua dokumen. Sebuah kata wi dalam c (R, S) jika memenuhi kondisi berikut :

Kesimpulan

Kami mempertimbangkan bagaimana membangun Mekanisme Deteksi Salin yang akurat dan efisien untuk mendeteksi salinan dokumen ilegal di Perpustakaan Digital. Kami menyajikan repositori registrasi dan struktur data persisten utama yang kami terapkan dalam SCAM, prototipe eksperimental kami. Struktur ini berguna untuk secara efisien menemukan deteksi yang tumpang tindih antara dokumen, sekaligus mempertahankan kemampuan untuk menskalakan hingga ratusan ribu dokumen. Kami menguraikan berbagai potongan primitif dan mempertimbangkan dampaknya pada ukuran kinerja kritis dan akurasi. Kami menyajikan hasil empiris pada masing-masing primitif chunking yang berbeda, dan menguraikan manfaat relatif dari primitif chunking tertentu atas yang lain dalam hal kinerja, dan akurasi. Kami menunjukkan bahwa kinerja dan akurasi sangat bervariasi untuk mekanisme chunking yang berbeda, sehingga penting untuk mengevaluasi dan memahami berbagai opsi chunking. Ini juga menggembirakan

bahwa kita memang bisa membangun mekanisme pendeteksian salinan yang dapat meningkatkan skala untuk repositori registrasi besar. Di masa depan, kami bermaksud meningkatkan repositori pendaftaran kami untuk mengkonfirmasi skalabilitas yang sebenarnya. Kami juga ingin lebih banyak bereksperimen dengan kata-kata berhenti, skema breakpoint hash yang berbeda, dan skema berdasarkan pohon PAT.