Aplikasi Pendeteksi Plagiarisme Dokumen dengan Algoritma Branch and Bound

Wilhelmus Andrian Tanujaya / 13513071
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13513071@std.stei.itb.ac.id

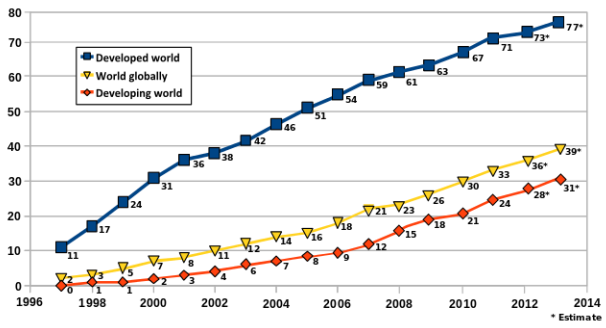
Makalah ini berisikan pemikiran mengenai pencarian kemiripan konten sebuah dokumen dengan dokumen lainnya untuk menentukan apakah dokumen yang diperiksa merupakan plagiarisme. Pencocokkan isi konten akan dilakukan dengan algoritma Branch and Bound.

Plagiarisme, penjiplakan, Branch and Bound, BFS, dokumen

I. PEMBUKAAN

Dunia internet sekarang sedang berkembang pesat. Internet dikembangkan sejak akhir tahun 1960 oleh ARPANET, *Advanced Research Projects Agency Network*. Dimulai dengan memanfaatkan teknologi *switch* yang dikembangkan oleh ARPANET, pada tahun 1970 internet mengalami kemajuan pesat dengan dikenalkannya *Transmission Control Protocol* dan *Internet Protocol* (TCP/IP). Perkembangan internet sangat pesat hingga mencapai bentuk yang kita kenal sekarang pada tahun 1990[1].

Hingga sekarang, pengunaan internet sudah meluas hingga ke mana-mana. Diperkirakan data yang ada pada internet adalah sebesar 1,2 Zettabytes atau setara dengan 1,3 miliar GB pada tahun 2010. Bahkan, data yang disimpan di internet sejak tahun 2009 hingga 2010 mencapai 62 persen[2].



Gambar 1. Jumlah Pengguna Internet di dunia per 100 orang
(Sumber : http://upload.wikimedia.org/wikipedia/commons/thumb/2/29/Internet_users_per_100_inhabitants_ITU.svg/650px-Internet_users_per_100_inhabitants_ITU.svg.png diakses pada 3 Mei 2015 pukul 04.33 WIB)

Dengan banyaknya data yang disimpan di internet, sebuah masalah kemudian muncul ke permukaan, dan dalam waktu yang singkat, yaitu masalah keorisinilan ide. Banyak sekali kasus penjiplakan yang terjadi di internet dan umumnya tidak dapat dikendalikan. Beberapa kasus penjiplakan bahkan menyeret nama-nama besar seperti wakil presiden ke-47 Amerika Serikat, Joseph Robinette "Joe" Biden Jr, dan presiden Rusia, Vladimir Putin[3].

Semenjak maraknya internet di dunia, banyak usaha untuk memberantas plagiarisme bermunculan. Beberapa situs yang dianggap menyebarkan konten plagiat ditutup secara serentak. Di antaranya adalah website yang cukup populer di masa itu, Megaupload.com oleh FBI pada Januari 2012[4].

Selain dengan penutupan situs-situs yang dianggap sarat akan konten plagiat, banyak jasa pengecekkan dokumen plagiat yang beredar di internet sekarang. Tindakan ini ada untuk mencegah plagiarisme terjadi, terutama di kalangan mahasiswa yang sering berurusan dengan dokumen dan dituntut keorisinal dokumennya.

Penulis pun menjadi tertarik untuk membahas masalah plagiarisme ini. Pendeteksian plagiarisme yang dimaksudkan di sini adalah mencocokkan kalimat-kalimat pada dokumen dengan kalimat-kalimat yang ada pada dokumen-dokumen pembanding. Bila ditemukan kalimat dengan kata dan struktur yang sama persis, kalimat tersebut akan ditandai, dan pengguna aplikasi diberitahu mengenai dokumen yang mungkin menjadi sumber jiplakan.

II. TEORI DASAR

A. Plagiarisme

Plagiarisme merupakan tindakan penjiplakan tanpa menyebutkan sumber kutipan yang digunakannya, seolah-olah apa yang dijiplak tersebut adalah pemikiran sendiri. Plagiarisme dapat dilakukan pada tulisan, pembicaraan, lagu dan ide. Sumbernya bermacam-macam, antara lain laman web, buku, lagu, acara TV, karya lukis, atau apapun[5].

Meskipun kontennya sama, terkadang sebuah karya tidak dapat dicap sebagai plagiarism bila memenuhi beberapa kriteria. Kriteria yang dimaksud adalah, karya merupakan sebuah *Common Knowledge* atau pengetahuan umum. Kata-kata yang ada pada sebuah karya, dalam karya yang sama tidak pernah akan dicap sebagai plagiat.

Berbeda dengan kata-kata sebuah karya yang terdapat pada karya lain, meskipun kepemilikan karya keduanya adalah sama. Selain itu, clip art yang ada di komputer juga bukan merupakan sesuatu yang dapat dicap plagiat bila seorang penulis menggunakannya di publik[6].

Di samping alasan-alasan di atas, sebuah karya juga dapat terhindar dari cap plagiarisme karena konten yang dimilikinya masuk ke dalam kategori abu-abu. Konten yang masuk ke dalam kategori ini adalah penemuan ilmiah terbaru, bahkan yang sudah masuk ke dalam headline berita. Konten abu-abu lainnya adalah hasil percakapan dengan orang lain, atau pun pengetahuan yang sebenarnya tidak umum namun dianggap umum oleh beberapa pihak[6].

Adapun cara untuk mengutip tanpa terkena cap plagiarisme adalah dengan cara memberikan sumber kutipan langsung pada tulisan, atau dikenal dalam aturan MLA(*Modern Language Association*) sebagai *In-text citations.* Cara lain adalah dengan memberikan sumber kutipan pada catatan kaki dokumen. Selain keduanya, seorang penulis dokumen juga dapat mencantumkan sumber-sumber konten yang disalinnya dalam Pustaka dokumennya[7].

B. Breadth First Search

Breadth First Search atau sering disingkat menjadi BFS merupakan sebuah algoritma traversal graf yang mengunjungi simpul-simpulnya secara sistematik melebar. Algoritma ini dapat diterapkan pada graf statis (graf yang telah terbentuk sebelum pencarian) maupun pada graf dinamis (graf yang terbentuk saat proses pencarian dilakukan) [8].

Pencarian sistematik melebar, artinya pencarian solusi permasalahan akan dilakukan pada simpul-simpul yang bertingkat sama terlebih dahulu. Bila tidak ditemukan solusi, pencarian dilanjutkan pada semua simpul pada tingkat berikutnya. Demikian pencarian akan terus berlanjut hingga sebuah solusi ditemukan[8].

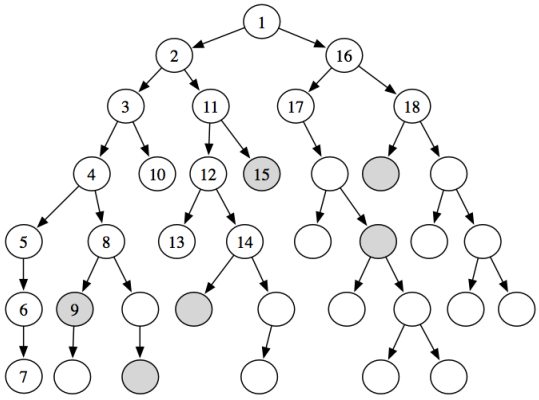
BFS dapat dinilai melalui terpenuhinya beberapa aspek. Aspek-aspek yang dapat dijadikan penilaian dalam penyelesaian masalah dengan algoritma BFS di antaranya adalah *completeness*, yaitu penjaminan solusi ditemukan. Selain itu ada aspek *Optimality* yaitu jaminan hasil yang optimal, *Time Complexity dan Space Complexity* yang merupakan waktu dan ruang yang dibutuhkan untuk pencarian[8].

Dari aspek-aspek yang disebutkan di atas, BFS memiliki hasil analisis sebagai berikut. *Completeness* akan selalu terpenuhi selama jumlah *branch* yang akan diperiksa diketahui. *Optimality* tidak dijamin pada BFS sebab hanya pada kasus langkah = biaya *optimality* dapat tercapai. Kompleksitas waktu yang dihasilkan adalah $O(b^d)$ yang tidak terlalu buruk. Sayangnya, kompleksitas ruang juga membutuhkan $O(b^d)$, yang tergolong buruk untuk kompleksitas ruang[8].

C. Branch and Bound

Algoritma Branch and Bound adalah algoritma yang memecahkan sebuah masalah dengan cara memberi bobot pada setiap simpul yang mungkin menjadi simpul solusi persoalan terlebih dahulu. Setelah diberi bobot, langkah berikutnya adalah memilih simpul yang akan dibangkitkan dengan acuan bobot simpul terendah. Pemilihan simpul yang dikunjungi berikutnya akan selalu dilakukan pada simpul yang memiliki bobot terendah, hingga hasil yang diinginkan dicapai[8].

Pada umumnya, masalah yang diselesaikan dengan algoritma Branch and Bound adalah masalah yang memiliki simpul solusi yang tidak diketahui keberadaannya; bagaimana lintasan untuk mencapainya dari titik awal. Akan tetapi, persoalan yang diketahui simpul akhirnya pun dapat dicapai dengan Branch and Bound, seperti permainan 8-puzzle[8].



Gambar 2. Ilustrasi Algoritma Branch and Bound
(Sumber : http://artint.info/figures/ch03/sgraph_bb.png diakses pada 3 Mei 2015 pukul 04.33 WIB)

Inti dari Branch and Bound sebenarnya adalah tidak memeriksa simpul-simpul yang tidak dapat mencapai solusi. Dengan demikian, algoritma pencarian seperti BFS dan DFS dapat dibatasi untuk mendapatkan keuntungan performansi[7].

III. APLIKASI ALGORITMA BRANCH AND BOUND PADA PENDETEKSI PLAGIARISME

Pada aplikasi pendeteksi plagiarisme, dokumen yang ingin diperiksa dipecah-pecah per-kalimat. Kita sebut kalimat-kalimat ini sebagai **kalimat-kalimat banding**. Dokumen - dokumen pembanding lainnya pun dipecah menjadi kalimat-kalimat. Kalimat-kalimat dari dokumen pembanding ini kita sebut saja sebagai **kalimat-kalimat pembanding**.

Kata pertama pada kalimat banding kemudian dicari pada tiap kalimat banding. Ketika ditemukan, upa-kalimat pembanding tersebut, yaitu upa-kalimat setelah kata pada kalimat banding ditemukan, dibuat tipe bentukannya dan kita beri nama simpul. Simpul mengandung atribut lain berupa angka yang menyatakan nomor dari dokumen manakah kalimat tersebut berasal. Selain itu, simpul juga mengandung sebuah angka lain yang menunjukkan banyak kata pada kalimat tersebut. Tipe bentukkannya dapat kita

misalkan seperti:

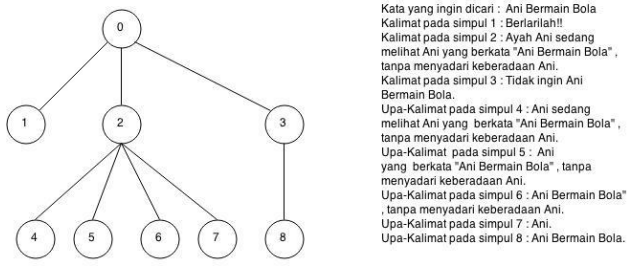
```
Struct simpul contains
      String Kalimat;
      Int no_Dokumen;
      Int n_Kata;
end
```

Setelah diubah ke dalam bentuk simpul-simpul, semua simpul kemudian disimpan dalam sebuah larik yang akan digunakkan untuk pencocokkan.

Pohon pencarian kemudian dibentuk dengan tingkat 1 adalah semua simpul yang ada. Artinya, kalimat pertama pada dokumen akan dicocokkan dengan semua kalimat yang ada pada simpul.

Pada tingkat 1, simpul-simpul yang tidak mungkin menghasilkan solusi akan dimatikan terlebih dahulu. Simpul yang dimaksudkan adalah simpul-simpul yang mengandung banyak kata lebih kecil daripada jumlah kata yang dimiliki oleh kalimat banding. Kemudian, kata pertama pada kalimat banding akan dicari pada simpul-simpul yang masih hidup. Apabila tidak ditemukan, simpul tersebut juga dibunuh.

Pada tingkat berikutnya, semua simpul yang masih hidup akan memiliki simpul-simpul anak yang merupakan upa-kalimat. Upa-kalimat yang dijadikan simpul adalah upa-kalimat dari kata pertama kalimat banding yang ditemukan pada kalimat pembanding tersebut (simpul orang tua). Pencarian kemudian dilakukan lagi dengan cara yang sama dengan pencarian pada tingkat 1, hingga hasil ditemukan pada tingkat-n (banyak kata pada kalimat banding) atau hasil tidak ditemukan. Berikut adalah ilustrasi pembentukan pohon yang dimaksud.



Gambar 3. Ilustrasi Pembentukan Pohon Status

Berikut adalah pseudocode aplikasi pendeteksi plagiarisme ini.

```
TIPE BENTUKAN
Struct simpul contains
  String Kalimat;
Int no_Dokumen;
  Int n_Kata;
end
```

```
DEKLARASI
S_Awal = simpul[];

//SETTING KONDISI AWAL
for each Documents
  for each Kalimat
    SimpulAwal.add(simpul(Kalimat))
    end for
  end for
```

```
ALGORITMA
function Banyak_Kata(Kalimat K)
//mengembalikan banyak kata yang
dimiliki oleh K

function FindPlagiat(simpul[] S,
Kalimat K)
  q : antrian;
  int found = -1;
  for each simpul in S
    if S.n_Kata >= banyak_Kata(K)
      q.add(S)
    end if
  end for
  while not q.Kosong do
    q.pop(&A) //A adalah sebuah tipe
                bentukan simpul
    yes = boolean;
    for each K.kata(i) found in A
      yes = true
      for (int j = i-1; j>=0; j--)
         if K.kata(j) = A.Kalimat(
           indeksfound -(i-j))
        end if
        else
          yes = false;
        end else
       endfor
        //BAGIAN BOUNDING
        if (yes)
          q.add(substring A.Kalimat
            (indeks found, A.length))
        end if
        // END OF BOUNDING
    end for
    i++;
    if (i == K.length)
      found = A.n_Kata
    end if
  end while

  return found;

end function;
```

Fungsi di atas akan diulang sebanyak n buah kalimat yang dimiliki oleh dokumen yang akan dideteksi konten plagiatnya. Setiap hasil yang dikeluarkan, yaitu int found, menunjukkan kesamaan kalimat pada dokumen dengan indeks found. Bila indeks found = -1, artinya tidak didapat dokumen yang memiliki konten serupa. Perhatikan pada algoritma pengisian simpul ke antrian, ada kemungkinan

beberapa simpul tidak memiliki anak. Maka, pencarian tidak dilanjutkan ke simpul berikutnya. Di sinilah bound digunakan.

Dengan menggunakan algoritma di samping, kompleksitas pencarian dapat ditekan seminim mungkin. Kondisi terbaik adalah ketika hanya ada satu kata dari kalimat-kalimat pembanding yang sesuai dengan kata pertama dari kalimat banding. Dengan demikian, pada tingkat 1, terdapat n kata yang perlu dibandingkan. Akan tetapi, pada tingkat 2, hanya 1 kata lagi yang dibandingkan, hingga akhirnya mencapai kata terakhir dari kalimat banding. Kompleksitasnya adalah O(n), yang didapat dari n(jumlah simpul pada tingkat 1), dikalikan 1 sebanyak m-1(jumlah kata pada kalimat banding) kali.

IV. KELEMAHAN ANALISIS

Pencarian solusi dengan algoritma BFS terkenal sangat boros akan ruang. Kompleksitas ruang untuk BFS pada umumnya adalah $O(b^d)$. Dengan menggunakan fungsi bounding pada pemasukkan elemen tingkat berikutnya ke dalam antrian, kompleksitas ruang dan kompleksitas waktu dapat dikurangi hingga cukup signifikan. Sayangnya, pada kasus terburuk, misalnya ketika sebuah kata yang ingin dicari, hanya berbeda pada akhir kalimat, kompleksitas ruang akan tetap mencapai $O(b^d)$.

Sebaiknya bagian *bounding* lebih dikembangkan lagi, agar pencocokkan kata tidak hanya dilakukan pada kata pertama, melainkan dilakukan dari belakang juga. Dengan demikian, masalah di atas dapat diselesaikan.

V. KESIMPULAN

Teori Branch and Bound dapat digunakan untuk mendeteksi konten plagiat pada sebuah dokumen. Dengan memanfaatkan algoritma ini, pencarian dokumen yang memiliki konten yang mirip dapat dilakukan dengan cepat, sangat bermanfaat bila digunakan oleh orang yang perlu memeriksa banyak dokumen, contohnya guru atau dosen. Bila dikembangkan lebih lagi, harapannya adalah pendeteksi plagiarisme menjadi semakin mudah untuk dijangkau oleh orang-orang, dan tingkat plagiarisme yang terjadi dapat ditekan.

VI. UCAPAN TERIMA KASIH

Pertama – tama saya ucapkan terima kasih kepada Tuhan Yang Maha Esa atas rahmat-Nya makalah ini dapat saya selesaikan. Tak lupa saya ucapkan terima kasih kepada orang tua saya yang telah membesarkan dan mendidik saya, serta menyekolahkan saya hingga ke jenjang perguruan tinggi ini. Ucapan terima kasih juga saya sampaikan kepada Pak Rinaldi Munir dan Ibu Nur Ulfa Maulidevi sebagai dosen Mata Kuliah IF2211 Strategi Algoritma atas pengetahuan tentang teori-teori Strategi Algoritma yang telah diajarkan kepada saya, sehingga saya bisa menyelesaikan makalah ini.

REFERENSI

[1] Ask History, *Who Invented the Internet* ,http://www.history.com/news/ask-history/who-invented-the-internet, *13 Desember, 2013, diakses pada 3 Mei 2015*

[2] GreenHaw, Andy. *How Big is The Internet?* https://andygreenhaw.wordpress.com/tag/how-much-data-exists-online/, 2010, diakses pada 3 Mei 2015.

[3] Smith,Sarah. *10 High profile plagiarism cases*. *http://www.politico.com/gallery/2014/07/10-high-profile-plagiarism-cases/001951-027782.html*, 7 Juni 2014, diakses pada 3 Mei 2015.

[4] *FBI shuts down Megaupload.com, Anonymous shut down FBI.* http://www.news.com.au/technology/fbi-shuts-down-megauploadcom-charges-seven-with-online-piracy/story-e6frfro0-1226249114650, Januari 2012, diakses pada 3 Mei 2015.

[5] http://www.lib.usm.edu/legacy/plag/whatisplag.php, diakses pada 3 Mei 2015.

[6] Washington State University Library, http://www.wsulibs.wsu.edu/library-instruction/plagiarism/what, diakses pada 4 Mei 2015

[7] https://owl.english.purdue.edu/owl/section/2/11/, diakses pada 3 Mei 2015.

[8] Rinaldi Munir, IF2211 Strategi Algoritma, http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/stmik.htm diakses pada 4 Mei 2015.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 4 Mei 2015

Wilhelmus Andrian Tanujaya, 13513071