Deteksi *Cyberbullying* pada Twitter dengan Algoritma Knuth Morris-Pratt dan Boyer-Moore

Kayla Namira Mariadi - 13522050
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13522050@std.stei.itb.ac.id

***Abstract*— *Cyberbullying* merupakan masalah serius di platform media sosial, seperti Twitter. Media sosial menyediakan peluang komunikasi antar pengguna, namun juga meningkatkan kerentanan dan aktivitas kejahatan siber. Sebuah survei terbaru melaporkan bahwa *cyberbullying* menjadi masalah yang berkembang di kalangan pengguna media sosial. Makalah ini berfokus pada deteksi *cyberbullying* di media sosial Twitter dengan bantuan algoritma pencocokan string, yaitu algoritma Knuth Morris Pratt dan Boyer Moore. Sistem ini dirancang untuk mengidentifikasi bahasa dan pola kata abusif dalam cuitan secara efisien. Solusi yang diusulkan diimplementasikan dan diuji pada dataset twitter. Hasil penelitian ini diharapkan dapat membantu dalam upaya mitigasi *cyberbullying* di media sosial.**

Keywords*— *cyberbullying; string-matching; KMP;Boyer-Moore

I. Pendahuluan

Perundungan atau *bullying* bukanlah fenomena baru di dunia ini. *Bullying* biasanya terbatas pada tempat, waktu, dan cenderung dapat diprediksi, sementara perundungan siber atau *cyberbullying* dapat terjadi kapan saja dan di mana saja. Dengan keberadaan internet dan media sosial yang semakin meluas seperti Twitter, Facebook, Instagram, serta aplikasi pesan seperti WhatsApp, Instagram, Telegram, dan banyak aplikasi lainnya memungkinkan komunikasi dengan siapa saja tanpa memandang tempat dan waktu.

Media sosial memiliki dua sisi: sisi positif di mana penggunanya dapat berbagi informasi yang bermanfaat dan membangun hubungan sosial, dan sisi negatif di mana terdapat peningkatan risiko bagi anak-anak dengan pesan-pesan yang mengancam, *cyberbullying*, dan *cyberstalking*. Dalam beberapa tahun terakhir, penggunaan jejaring sosial oleh individu dan perusahaan meningkat secara drastis. Hal ini disebabkan oleh pertumbuhan cepat perangkat teknologi, fasilitas internet, dan lainnya. Internet memiliki kecepatan dan kemampuan untuk mentransfer data yang terutama digunakan untuk tujuan komunikasi dan di sisi lain membuka pintu bagi oknum tak bertanggung awab untuk terlibat dalam kejahatan.

Seiring dengan pesatnya perkembangan teknologi, kejahatan siber juga meningkat secara eksponensial. Sebagian besar kejahatan siber menargetkan informasi tentang individu, pemerintah, atau korporasi. *Cyberbullying* adalah sisi gelap dari dunia teknologi, terutama di media sosial. Oleh karena itu, diperlukan deteksi *cyberbullying*. Dalam makalah ini, penulis mengusulkan cara untuk mendeteksi *cyberbullying* di media sosial Twitter menggunakan algoritma pencocokan string.

II. Landasan Teori

A. Pencocokan String

Pencocokan string biasanya melibatkan dua string: teks T[1...n] yang merupakan string utama, dan pola/*pattern* P[1...m] yang merupakan string yang akan dicocokkan dengan string utama, dengan syarat $m \leq n$. Secara umum, ada dua jenis pencocokan string: Pencocokan String Tepat (*Exact String Matching*) dan Pencocokan String Mendekati (*Approximate String Matching*). *Exact String Matching* bertujuan menemukan kemunculan pola pada teks secara sempurna. Pencocokan String Mendekati memungkinkan pencarian yang tidak akurat berdasarkan aplikasi tertentu, misalnya Algoritma Levenshtein dan Hamming.

Berdasarkan jumlah pola, pencocokan string memiliki dua klasifikasi: Pencocokan String Pola Tunggal dan Pencocokan String Pola Ganda. Dalam Pencocokan String Pola Tunggal, satu pola dicari dalam teks, sedangkan dalam Pencocokan String Pola Ganda, beberapa pola dicari dalam teks. Berdasarkan urutan pencarian, pencocokan string memiliki empat klasifikasi: pencocokan dari kiri ke kanan, pencocokan dari kanan ke kiri, pencocokan dengan urutan tertentu, dan pencocokan tanpa urutan tertentu.

Strategi pencocokan string yang paling dasar dan konvensional adalah Algoritma Brute Force yang mempertimbangkan semua kemungkinan kasus dan menggeser hanya satu tempat ke kanan bahkan jika terjadi kesesuaian atau ketidaksesuaian. Algoritma ini juga dikenal sebagai Pendekatan Naif. Untuk menghindari banyaknya perbandingan dalam Algoritma Brute Force, pada tahun 1970 algoritma Morris dan Pratt. Algoritma ini didasarkan pada pemrosesan pola sebelum pencocokan dan membandingkan karakter dari kiri ke kanan. Jika terjadi ketidaksesuaian, beberapa karakter dilewati, tidak digeser satu per satu seperti Brute Force.

Pada tahun 1977, Boyer dan Moore juga mengusulkan algoritma yang membandingkan karakter dari kanan ke kiri. Ada banyak algoritma pencocokan string pola ganda yang telah diusulkan dalam beberapa dekade terakhir seperti: Pada tahun

1975, algoritma Aho-Corasick yang dipresentasikan oleh Alfred V. Aho dan Margaret J. Corasick, yang membangun automata untuk pola pada fase pra-pemrosesan. Commentz Walter mengusulkan algoritma yang didasarkan pada algoritma Aho-Corasick dan Boyer-Moore. Algoritma Rabin Karp juga digunakan untuk mencari beberapa pola.

Beberapa aplikasi pencocokan string adalah editor teks dalam mesin komputasi, *query* basis data, bioinformatika, pencocokan dua dimensi, sistem deteksi intrusi jaringan, pencocokan pola jendela lebar (pencocokan string besar), pengambilan konten musik, pemeriksa sintaks bahasa, pemeriksa ejaan *ms word*, pencocokan urutan DNA, perpustakaan digital, mesin pencari, dan banyak lagi aplikasi lainnya.

B. Algoritma Knuth Morris Pratt

Algoritma KMP adalah algoritma yang dibuat oleh James H. Morris yang kemudian ditemukan kembali oleh Donald Knuth tidak lama setelahnya. Morris dan Vaughan Pratt juga mempublikasikan laporan tentang algoritma ini pada tahun 1970. Pada tahun 1977, Knuth, Morris, dan Pratt mempublikasikan algoritma ini bersama-sama. Cara kerja algoritma ini sama dengan brute force dalam hal urutan pencarian, yaitu dimulai dari kiri ke kanan, namun penentuan kapan penunjuk indeks yang sedang dicek akan digeser berbeda dan lebih teroptimasi.

Algoritma pencarian string KMP lebih teroptimasi dibandingkan brute force ketika terjadi ketidakcocokan karakter dengan menentukan di mana pencocokan berikutnya dapat dimulai, sehingga melewati pemeriksaan ulang karakter yang sebelumnya sudah dicocokkan.

Caranya adalah dengan menggunakan fungsi pinggiran b(k) yang merupakan ukuran prefix terbesar dari P[0..k] yang juga merupakan suffix dari P[1..k]. Keluaran dari b(k) ini merepresentasikan jumlah pergeseran maksimal yang bisa dilakukan agar tidak ada proses pencocokan yang sia-sia.

Misalkan panjang pola didefinisikan sebagai m dan panjang teks didefiniskan sebagai n, maka:

- Algoritma pembentukan fungsi pinggiran memiliki kompleksitas waktu O(m)
- Algoritma KMP utama memiliki kompleksitas waktu O(n).
- Total kompleksitas waktu algoritma KMP ini adalah O(m+n).

Selain menghindari pencocokan yang sia-sia, algoritma ini juga memproses teks secara maju sehingga cocok untuk memproses file yang diberikan dalam bentuk *stream* dan tidak perlu memeriksa kembali karakter yang sudah diperiksa sebelumnya. Namun, performa KMP akan lebih buruk apabila jumlah alfabet variatif, karena apabila ada ketidakcocokan pada awal karakter, algoritma ini akan menjadi lebih lambat, menyerupai Brute Force yang menggeser lokasi pencocokan satu per satu ke kanan. Salah satu kekurangan lain dari algoritma ini yaitu tidak mempertimbangkan karakter yang menyebabkan ketidakcocokan.

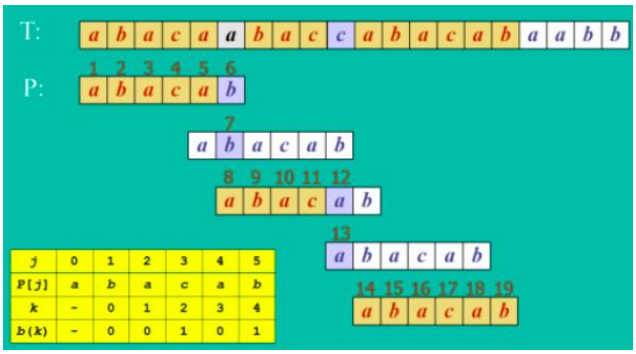

Fig. 1. Contoh penerapan algoritma KMP (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Makalah2021/Makalah-Stima-2021-K2%20(45).pdf)

C. Algoritma Boyer Moore

Algoritma Boyer-Moore adalah algoritma yang dikembangkan oleh Robert S. Boyer dan J. Strother Moore pada tahun 1977. Algoritma Boyer-Moore memanfaatkan dua teknik, yaitu:

- *Looking-glass*

  Teknik ini mencari pattern pada teks dengan pola terbalik, dari kanan ke kiri.

- *Character-jump*

  Lokasi pencocokan akan bergeser atau 'lompat' dengan cara yang berbeda, tergantung dengan keberadaan karakter yang *mismatch* pada pola:

  1. Jika lokasi karakter pada P berada sebelum lokasi pencocokan saat ini, geser P ke kanan untuk meluruskan kemunculan terakhir karakter pada P dengan T[i].
  2. Jika lokasi karakter pada P berada setelah lokasi pencocokan saat ini (tidak bisa menggeser P ke kanan), maka geser P sebanyak 1 karakter ke T[i+1].
  3. Jika P tidak mengandung karakter, geser P sampai P[0] dan T[i+1] sejajar.

Untuk menyimpan kemunculan terakhir atau *last occurrence* karakter pada algoritma Boyer-Moore, dibuat sebuah fungsi *Last Occurrence Function L(x)* yang berfungsi untuk mengembalikan indeks terbesar i yang membuat P[i] == x, atau -1 apabila tidak ada indeks yang memenuhi. Untuk setiap karakter pada pola, diaplikasikan fungsi ini dan biasanya disimpan dalam suatu array sebelum dilakukan pencocokan.

Algoritma ini, sama seperti Brute Force, akan lebih baik ketika jumlah huruf pada alfabet bervariasi, sehingga lebih cocok untuk mencocokkan teks berbahasa Inggris, misalnya, dibandingkan dengan teks biner.

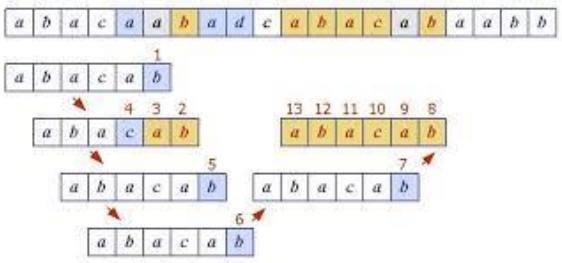

Fig. 2. Contoh penerapan algoritma BM (Sumber: https://stackoverflow.com/questions/53623770/understanding-boyer-moore-visually)

Misalkan panjang pola didefinisikan sebagai A dan panjang teks didefinisikan sebagai n, kompleksitas waktu untuk menghitung *last occurence* yaitu O(A), maka untuk kasus terburuk, kompleksitas waktunya adalah O(nm + A).

D. Regex

Regex (*Regular Expressions)* adalah metode yang digunakan untuk mencocokkan pola tertentu dalam teks. Regex banyak digunakan dalam pemrosesan teks karena kemampuannya dalam mencari, mengganti, atau memanipulasi string berdasarkan pola tertentu. Regex menggunakan karakter khusus untuk mendefinisikan pola pencarian yang dapat mencakup huruf, angka, atau karakter khusus lainnya. Beberapa contoh penggunaan regex dalam pemrosesan teks termasuk:

- **Pencocokan Pola**

  Mencari pola spesifik dalam teks, seperti alamat email atau nomor telepon.

- **Penggantian String**

  Mengganti bagian dari teks dengan string lain berdasarkan pola yang cocok.

- **Validasi Data**

  Memastikan bahwa teks memenuhi format tertentu, seperti format kode pos atau nomor kartu kredit.

- **Ekstraksi Data**

  Menarik informasi tertentu dari teks yang lebih besar, seperti ekstraksi tanggal dari teks dokumen.

Regex sangat berguna dalam konteks deteksi *cyberbullying* karena dapat digunakan untuk menemukan kata-kata kasar, pola ujaran kebencian, dan bentuk lain dari bahasa abusif dalam teks.

E. Cyberbullying

Cyberbullying adalah bentuk penindasan yang terjadi melalui media elektronik, khususnya media sosial. Ini melibatkan penggunaan teknologi untuk mengganggu, memalukan, atau mengintimidasi seroang individu. Beberapa bentuk umum cyberbullying meliputi ujaran kebencian, intimidasi, pemerasan, penyebaran informasi pribadi, atau penyebaran hoax.

Cyberbullying memiliki dampak yang signifikan terhadap korban, termasuk masalah kesehatan mental seperti depresi, kecemasan, dan dalam kasus ekstrem, mendorong tindakan bunuh diri. Oleh karena itu, deteksi dini dan intervensi sangat penting untuk mengurangi dampak negatif dari *cyberbullying*.

III. METODOLOGI

Akan digunakan algoritma KMP dan Boyer-Moore untuk mendeteksi apakah sebuah cuitan termasuk *cyberbullying* atau bukan. Kedua algoritma akan dibandingkan dan diberikan hasil ujinya untuk mengetahui algoritma mana yang paling efektif menjadi implementasi. Berikut adalah langkah pendeteksian *cyberbullying* pada twitter dengan algoritma KMP dan BM menggunakan bahasa python.

A. Pengumpulan Dataset

Dataset berasal dari data *cyberbullying* pada Twitter yang diunduh dari Kaggle. Dataset tersebut terdiri dari 47000 cuitan berbahasa Inggris yang dikategorikan berdasarkan tipe *cyberbullying*-nya:

- Umur;
- Etnis;
- Jenis kelamin;
- Agama;
- Jenis lain *cyberbullying*;
- Bukan *cyberbullying*

Dataset sudah seimbang, dengan jumlah data berjumlah sekitar 8000 untuk setiap kelas. Untuk simplifikasi, hanya akan diambil sampel sejumlah 50 dari seluruh dataset. Berikut adalah distribusi data per tipenya dari sampel yang diambil:

```
religion              11
ethnicity              9
gender                 9
other_cyberbullying    9
not_cyberbullying      7
age                    5
Name: cyberbullying_type, dtype: int64
```

Fig. 3. Distribusi tipe data

Berikut adalah cuplikan cuitan dari dataset beserta tipe *cyberbullying*-nya.

```
                                              tweet_text   cyberbullying_type  \
34799  I'm the same thing,I was too bullied in middle...                  age
44729  Somebody called me a white supremacist. Uncle ...            ethnicity
26410  however, it doesn't scale. i can use this to t...  other_cyberbullying
```

Fig. 4. Contoh Dataset

B. Pemrosesan Data

Sebelum melakukan pencocokan *string*, perlu dilakukan pembersihan data (data *cleansing/preprocessing)* dengan bantuan kakas python, pandas. Hål ini bertujuan untuk mempercepat proses pencocokan nantinya dan mereduksi *noise* data.

- *Casefolding*

  Pengubahan semua huruf menjadi huruf kecil untuk menyamakan format dan menghindari perbedaan karena kapitalisasi.

- Tokenisasi

  Tokenisasi adalah metode untuk memecah teks menjadi entitas kecil seperti kata atau frasa. Selama proses ini, elemen yang tidak diinginkan seperti tanda baca dihilangkan. Setiap token berguna untuk mengidentifikasi dan mengungkap pola dalam dokumen teks. Untuk melakukan tokenisasi, digunakan NLTK, kakas python untuk preproses data berupa teks.

- Penghapusan *stopwords*

  Stopwords adalah kata-kata yang paling sering digunakan dalam setiap dokumen seperti "is", "do", "was", "dan sebagainya. *Stopwords* perlu dihapus untuk meningkatkan efisiensi waktu dan ukuran performa karena kata-kata ini tidak memberikan makna yang signifikan pada kalimat. Dengan menghapus *stopwords*, semua kata yang tidak relevan diabaikan.

- Penghapusan karakter spesial, tagar, URL

  Karakter spesial dan URL ini dihapus karena tidak relevan dalam pendeteksian *cyberbullying*.

- Lematisasi

  Lematisasi adalah mereduksi setiap kata menjadi bentuk dasarnya. Proses ini dapat mengurangi dimensi data karena jumlah kata total menjadi lebih sedikit, sambil tetap mempertahankan makna aslinya. Dengan demikian, performa dan efisiensi model akan meningkat. Misalnya, *loving* menjadi *love*.

Berikut adalah perbandingan cuitan dari dataset sebelum dan setelah dibersihkan.

TABLE. 1. Perbandingan data sebelum dan setelah pemrosesan

No	Sebelum pemrosesan	Setelah pemrosesan
1	@BasicMountain it's hard for me to find good marvel schwag. i've got a nice print from target (lol), but DC sadly does better mechandising.	hard find good marvel schwag got nice print target lol dc sadly better mechandising
2	Curled up in my favorite chair. Apartment is warm, but cool breeze from the open window. Listening to the sounds of Oakland. I love it here.	curled favorite chair apartment warm cool breeze open window listening sound oakland love
3	vomit. #stopwadhwa2015 http://t.co/78tLSv56cs	vomit

C. *Pemilihan kata yang menjurus pada cyberbullying*

```
abusive_keywords = ['dumb', 'idiot']
```

Fig. 5. Contoh kata yang menjurus pada *cyberbullying*

Untuk simulasi pendeteksian *cyberbullying*, digunakan 2 kata di atas sebagai *pattern* yang akan dicocokkan dengan tiap cuitan.

D. *Proses Deteksi Cyberbullying dengan Algoritma KMP*

Setelah mempreproses data dan mengumpulkan kata-kata yang menjurus pada *cyberbullying*, akan diaplikasikan algoritma KMP untuk mencocokkan tiap cuitan atau *tweet* pada dataset dengan kata-kata tersebut (*pattern*). Jika sudah, akan ditampilkan semua cuitan beserta hasil deteksinya, ditampilkan pula tipe dari *cyberbullying*-nya sebagai pembanding.

IV. IMPLEMENTASI

Berikut adalah langkah pendeteksian *cyberbullying* pada Twitter dengan algoritma KMP menggunakan bahasa python.

A. *Persiapan Data*

Program akan membaca data yang disimpan dalam format csv terlebih dahulu, lalu menghapus data duplikat serta melakukan pembersihan data. Pembersihan data diimplementasikan pada fungsi *preprocess_tweet*. Program menggunakan Regex untuk menghapus tag HTML, URL (teks yang diawali dengan http), *mentions*(teks yang diawali dengan @), dan tagar (teks yang diawali dengan #). Regex juga, menghapus karakter non-alfabet (teks yang tidak mengandung karakter alfabet A-Z dan a-z). Kemudian, teks diubah menjadi huruf kecil dan ditokenisasi. *Stopwords* dihapus dengan menghapus kata-kata *stopword* dari token-token. Langkah terakhir yaitu lematisasi kata, fungsi kemudian mengembalikan teks yang telah digabung kembali menjadi suatu string.

Fungsi *preprocess_tweet* tersebut diaplikasikan ke tiap data cuitan pada dataset, lalu data-data diubah representasinya ke dalam suatu array bernama tweets. Berikut adalah implementasinya.

```
# Membersihkan data
def preprocess_tweet(text):
    # Menghapus tag HTML
    text = re.sub('<.*?>', '', text)

    # Menghapus URL, mention, dan hashtag dari
teks
    text = re.sub(r'http\S+', '', text)
    text = re.sub(r'@\S+', '', text)
    text = re.sub(r'#\S+', '', text)

    # Menghapus karakter non-alfabet dan mengubah
menjadi huruf kecil
    text = re.sub('[^a-zA-Z]', ' ', text).lower()

    # Tokenisasi teks
    words = nltk.word_tokenize(text)

    # Menghapus stopwords
    words = [w for w in words if w not in
stopwords.words('english')]

    # Menglematisasi kata
    lematizer = WordNetLemmatizer()
    words = [lematizer.lemmatize(w) for w in
words]

    # Menggabungkan kembali kata-kata menjadi
string
    text = ' '.join(words)
```

B. Implementasi Algoritma

Setelah data dipersiapkan dan dibersihkan, langkah selanjutnya adalah mengimplementasikan algoritma KMP (Knuth-Morris-Pratt) dan BM (Boyer-Moore) untuk mendeteksi keberadaan kata-kata yang menjerumus pada cyberbullying dalam tweet.

1.) KMP

```
def compute_border(pattern):
    pattern_length = len(pattern)
    border = [0] * pattern_length
    j = 0
    i = 1

    while i < pattern_length:
        if pattern[i] == pattern[j]:
            j += 1
            border[i] = j
            i += 1
        else:
            if j != 0:
                j = border[j - 1]
            else:
                border[i] = 0
                i += 1
    return border

def KMPSearch(pattern, text):
    M = len(pattern)
    N = len(text)
    border = compute_border(pattern)
    i = 0
    j = 0

    while i < N:
        if pattern[j] == text[i]:
            i += 1
            j += 1
            if j == M:
                return True
        elif j != 0:
            j = border[j - 1]
        else:
            i += 1
    return False
```

- Menghitung *border*: Fungsi compute_border merupakan fungsi *border*
- Pencarian Pola: Fungsi KMPSearch mencari pola dalam teks menggunakan array *border* untuk menghindari pemeriksaan karakter yang berulang.

2.) BM

```
def last_occurrence(pattern):
    length = len(pattern)
    last = [-1] * 128  # inisialisasi array
    for i in range(length):
        last[ord(pattern[i])] = i
    return last
```

```
def bm_match(pattern, text):

    last = last_occurrence(pattern)
    pattern_length = len(pattern)
    text_length = len(text)
    text_idx = pattern_length - 1

    if text_idx > text_length - 1:
        return False

    pattern_idx = pattern_length - 1
    while text_idx < text_length:
        if pattern[pattern_idx] ==
text[text_idx]:
            if pattern_idx == 0:
                return True
            text_idx -= 1
            pattern_idx -= 1
        else:
            lo = last[ord(text[text_idx])]
            text_idx += pattern_length -
min(pattern_idx, lo + 1)
            pattern_idx = pattern_length - 1
```

- Mencari array *last occurence*: Membuat array yang menyimpan index kemunculan terakhir tiap karakter pada pattern
- Pencarian Pola: Fungsi bm_match mencocokkan pattern dan text dengan Boyer-Moore.

C. Aplikasi Algoritma pada Dataset

Fungsi *detect_cyberbullying* menggunakan algoritma KMP dan BM untuk mendeteksi kata-kata kasar dalam tweet dan menerima 2 argumen; kata kasar atau *pattern* dan cuitan.

V. ANALISIS

A. Hasil Deteksi

Setelah menguji kedua algoritma, maka didapatkan hasil seperti berikut:

Fig. 6. Percobaan 1: Cuplikan perbandingan kedua algoritma

Fig. 7. Percobaan 2: Cuplikan perbandingan kedua algoritma

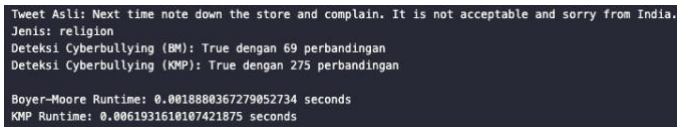

Fig. 8. Percobaan 3: Cuplikan perbandingan kedua algoritma

TABLE. 2. Perbandingan *runtime* kedua algoritma

Percobaan	KMP	BM
1	0.007s	0.005s
2	0.013s	0.007s
3	0.006s	0.0018s

Berdasarkan percobaan, mayoritas didapatkan *runtime* algoritma Boyer-Moore yang lebih cepat walaupun perbedaannya tidak signifikan.

TABLE. 3. Perbandingan jumlah pencocokan karakter kedua algoritma

Percobaan	KMP	BM
1	51	186
2	104	30
3	275	69

Berdasarkan percobaan, didapatkan jumlah pencocokan karakter pada algoritma Boyer-Moore lebih sedikit secara signifikan .

B. Analisis Performa

Dalam konteks pendeteksian *cyberbullying* pada cuitan, didapatkan *runtime* algoritma BM yang lebih cepat dibandingkan KMP. Perbedaan utama antara kedua algoritma ini terletak pada cara mereka menghindari perbandingan yang tidak perlu. Algoritma BM melakukan pencocokan dari kanan ke kiri serta menggunakan heuristik *last occurrence* yang memungkinkan banyak pergeseran saat terjadi ketidakcocokan, sehingga lebih cepat dalam percobaan kali ini, terutama pada teks yang panjang dan pola yang jarang muncul. Di sisi lain, KMP menghindari perbandingan berulang dengan menggunakan array berisi fungsi pinggiran, yang lebih efektif pada pola yang memiliki banyak pengulangan.Namun jika karakter yang tidak cocok berada pada awal pola, lompatan yang dilakukan oleh algoritma ini relatif kecil dibandingan dengan BM.

Pada percobaan 1 dan 2, selisih *runtime* kedua algoritma tidak terlalu berbeda jauh. Hal ini kemungkinan disebabkan oleh pola yang dicari memiliki distribusi yang tidak terlalu jarang atau terlalu sering sehingga kedua algoritma dapat menunjukkan kinerja yang serupa. Ukuran teks dan pola yang diuji juga dapat mempengaruhi kinerja. Algoritma BM dapat lebih efektif pada cuitan dengan alfabet beragam. Begitu pula, pada teks yang memiliki banyak pengulangan, keunggulan KMP dalam menghindari perbandingan berulang juga bisa tidak terlalu signifikan.

VI. SIMPULAN

Pattern matching adalah proses mencari kemunculan atau pencocokan dari suatu pola (*pattern*) dalam suatu teks atau data. Beberapa algoritma yang dapat digunakan dalam pattern matching yaitu algoritma Knuth-Morris Pratt dan algoritma Boyer-Moore. Dari percobaan, dapat disimpulkan bahwa secara rata-rata algoritma Boyer Moore memiliki performa *runtime* yang lebih efisien jika dibandingkan dengan Knuth Morris Pratt dalam pendeteksian teks *cyberbullying* pada Twitter, walaupun perbedaannya tidak signifikan. Sedangkan untuk jumlah perbandingan karakter, BM memiliki jumlah yang lebih sedikit secara signifikan. Algoritma BM secara rata-rata dapat menyelesaikan pencarian sedikit lebih cepat dibandingkan algoritma KMP karena melakukan banyak pergeseran atau lompatan saat terjadi ketidakcocokan. Selain itu, kecepatan eksekusi kedua algoritma juga dipengaruhi oleh ukuran cuitan dan pola kata kasar yang diuji.

VIDEO LINK YOUTUBE

Penjelasan mengenai makalah berikut dapat dilihat pada link berikut: https://youtu.be/kpL-zzQFHgw

UCAPAN TERIMA KASIH

Penulis mengucapkan puji dan syukur kepada Tuhan Yang Maha Esa atas berkah dan rahmatnya sehingga makalah ini yang berjudul "Deteksi *Cyberbullying* pada Twitter dengan Algoritma Knuth Morris-Pratt dan Boyer-Moore" dapat diselesaikan dengan baik dan lancar Selain itu, penulis juga ingin menyampaikan penghargaan yang setinggi-tingginya kepada semua pihak yang telah membantu dalam proses penyelesaian makalah ini, antara lain:

1. Dr. Nur Ulfa Maulidevi, S.T, M.Sc. selaku dosen pengampu mata kuliah IF2211 Strategi Algoritma K02.

2. Dr. Ir. Rinaldi Munir, M.T. yang telah memberikan kontribusi melalui buku dan materi yang diacu dalam makalah ini.

3. Semua individu atau kelompok yang telah berkontribusi dalam pembuatan makalah ini.

REFERENCES

[1] Andrew Davidson. 2006. "Pattern Matching" dalam Munir, Rinaldi (E.d.). http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2014-2015/Pencocokan%20String%20(2015).ppt (diakses pada 10 Juni 2024).

[2] J. Wang, K. Fu, C.T. Lu, "SOSNet: A Graph Convolutional Network Approach to Fine-Grained Cyberbullying Detection," Proceedings of the 2020 IEEE International Conference on Big Data (IEEE BigData 2020), December 10-13, 2020.

[3] H.Hosseinmardi, S. A. Mattson, R. I. Rafiq, R. Han, Q. Lv and S. Mishra, "Detection of Cyberbullying Incidents on the Instagram Social Network," Association for the Advancement of Artificial, pp. 5-6, 2015.

[4] G. Xiang, B. Fan, L. Wang, J. I. Hong and C. P. Rose, "Detecting Offensive Tweets via Topical Feature Discovery over a Large Scale Twitter Corpus," 2012.

[5] A. F. Hidayatullah, A. A. F. Yusuf, K. P. Juwairi and A. N. R. Nayoan, "Identifikasi Konten Kasar pada Tweet Bahasa Indonesia," Jurnal Linguistik Komputasional, vol. 2, no. 1, p. 3, 2019.

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, 12 Juni 2024

Kayla Namira Mariadi
13522050