Pencocokan String dalam Data Scraping

Naufal Putra Pamungkas - 13516110
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Bandung, Indonesia
13516110@std.stei.itb.ac.id

***Abstrak*—Data scraping adalah sebuah teknik perangkat lunak yang mengambil informasi dari sebuah halaman web. Salah satu proses dalam data scraping adalah mengambil informasi tertentu dari HTML sebuah halaman. Untuk mempermudah pengambilan informasi dari sebuah HTML, seseorang dapat membuat program untuk membantunya. String matching adalah sebuah algoritma untuk mencari lokasi di mana sebuah pattern ditemukan di sebuah teks. Makalah ini akan membahas implementasi string matching untuk membantu data scraping, terutama penggunaan string matching untuk mencari sebuah atau semua tag tertentu dari sebuah HTML, termasuk mencari tag dengan atribut tertentu.**

Kata Kunci – String Matching, HTML, Scraping

I. PENDAHULUAN

Seiring berkembangnya zaman, kebutuhan manusia semakin terbantu dengan teknologi internet. Banyak aktivitas dan informasi manusia zaman sekarang tersedia internet. Beberapa aktivitas dan informasi tersebut adalah jual-beli online, informasi olahraga, *booking* tiket, permainan (*video games*), dan masih banyak lagi. Banyaknya aktivitas manusia yang dilakukan di internet akan menyebabkan banyaknya data yang disetor di internet. Oleh karena itu, kebutuhan untuk mengekstrak data dari internet semakin penting agar data-data tersebut dapat diolah menjadi data baru. Misalnya, dengan maraknya jual-beli online, maka untuk membuat data penjualan suatu produk perlu mengekstrak data dari hasil penjualan online juga agar data penjualan tersebut lebih akurat, tidak hanya penjualan dari toko konvensional saja.

Mengekstrak data-data tersebut dari sebuah web disebut dengan data scraping. Mengekstrak suatu data dari web dapat dilakukan dengan mudah menggunakan sebuah *Application Programming Interface (*API) atau bisa juga dilakukan secara manual dengan cara me-*request* html dari web yang diinginkan, lalu mengekstrak data yang diinginkan dari html tersebut secara manual. Tidak semua website menyediakan API untuk memberikan data secara otomatis, oleh karena itu kemampuan mengekstrak data secara manual tetap diperlukan. Untuk mempercepat proses mengekstrak data secara manual, seseorang dapat membuat sebuah program yang secara otomatis mencari data yang ingin diekstrak. Program yang dibuat dapat memanfaatkan algoritma string matching untuk mempercepat pencariannya.

II. STRING MATCHING

String matching adalah sebuah proses atau algoritma untuk mencari lokasi (atau indeks) dimana sebuah string (atau beberapa string yang biasa disebut *pattern*) ditemukan dalam sebuah teks. Walaupun string matching hanya menemukan lokasi sebuah string, aplikasi dari algoritma ini sangatlah banyak, misalnya mesin pencari, pendeteksi intrusi jaringan, pencocokan DNA dan masih banyak lagi. Terdapat banyak jenis algoritma string matching, beberapa diantaranya adalah Knuth-Moriss Pratt Algorithm dan Boyer Moore.

Contoh *String Matching:*

Pattern: lompat

Teks: Rubah coklat melompati anjing malas

Hasil: 15-20 (Rubah coklat me**lompat**i anjing malas)

Sebelum membahas algoritma string matching, terdapat istilah pada string yang perlu diketahui, yaitu *prefix* dan *suffix*. Jika terdapat string X dengan panjang N, maka *prefix* dari string tersebut adalah substring X[0..k], dan *suffix* dari string tersebut adalah substring X[k..N-1] dengan k adalah nilai diantar 0 dan N-1.

A. Boyer-Moore Algorithm

Algoritma Boyer-Moore adalah algoritma string matching yang dikembangkan oleh Robert S. Boyer dan Strother Moore. Algoritma ini menggunakan teknik *looking-glass*, yaitu mencari pattern pada teks dengan pencocokan dimulai dari akhir pattern ke awal (Bukan akhir teks). Algoritma ini juga melakukan *preprocess* pada pattern untuk melakukan *character-jump* agar mempercepat proses pencarian. Preprocess ini mencatat indeks terakhir suatu alphabet ditemukan pada pattern untuk semua alphabet.

Boyer-Moore menggunakan teknik *character-jump*, yaitu jika ditemukan ketidak-cocokan karakter pada Teks[i] dan Pattern[j] dimana Teks[i] adalah X, pattern akan bergerak sesuai dengan salah satu dari tiga keadaan berikut:

1. Jika Pattern mengandung X, maka geser Pattern agar Teks[i] sejajar dengan X yang terakhir ditemukan pada P.
   Contoh:
   Teks: ###XABC###
   Pattern: XCBA
   Menjadi,
   Teks: ###XABC###
   Pattern: XCBA

2. Jika pattern mengandung X namun berada di kanan dari indeks dimana pattern mengalami ketidak-cocokan, maka geser pattern sebanyak satu karakter terhadap Teks
   Contoh:
   Teks: ###XAXC###
   Pattern: CBAX
   Menjadi,
   Teks: ###XAXC###
   Pattern: CBAX

3. Jika keadaan 1 dan 2 tidak terpenuhi, yaitu Pattern tidak mengandung X, maka geser pattern agar Pattern[0] sejajar dengan Teks[i+1]
   Contoh:
   Teks: ###XABC###
   Pattern: DCBA
   Menjadi,
   Teks: ###XABC###
   Pattern: DCBA

Teknik *character-jump* ini dilakukan agar program tidak perlu mencocokan semua substring dari teks dengan pattern (seperti *brute-force)*, namun program dapat dengan pintar menggerakan pattern untuk melewati proses pencocokan dengan substring yang sudah pasti tidak cocok.

B. *Knuth-Morris-Pratt Algorithm*

Algoritma KMP melakukan pencocokan string dari kiri ke kanan mirip seperti brute force, namun algoritma ini melakukan pergeseran yang lebih efisien dibanding brute force, yaitu melakukan pergeseran pattern ketika terjadi ketidakcocokan string sedemikian rupa agar melewati proses pencocokan yang sia-sia (pasti tidak cocok). Hal ini dapat diperoleh dengan cara menggeser pattern sebesar selisih panjang pattern dan panjang prefix Pattern[0..j-1] terpanjang yang sama dengan suffix Pattern[1..j-1], dimana j adalah indeks pada pattern yang tidak cocok dengan bagian teks.

Contoh:
Teks: ABAABX
Pattern: ABAABA

Ketidakcocokan terjadi pada j=5. Maka cari panjang prefix terpanjang dari ABAAB yang sama dengan suffixnya, yaitu 2 (AB). Maka, pattern akan digeser sebanyak 5-2=3 karakter menjadi:

Teks: ABAABX
Pattern: ABAABA

Proses pencocokan akan dimulai kembali dari Pattern[2]. Pergeseran ini sangat menghemat pencocokan karena melewati pencocokan yang mubazir. Untuk mendukung pencarian prefix dan suffix yang sudah dijelaskan sebelumnya, KMP akan melakukan preprocessing pada pattern. Preprocessing ini digunakan untuk menghitung dan mencatat terlebih dahulu panjang prefix terpanjang dari Pattern[0..j-1] yang sama dengan suffix Pattern[1..j-1]. Berikut contoh preprocessing dari sebuah pattern:

Pattern: ABAABA

j	0	1	2	3	4	5
Pattern[j]	A	B	A	A	B	A
F(k)	-	0	0	1	1	2

Dengan preprocessing yang menghasilkan tabel diatas, program hanya perlu melakukan kalkulasi pergeseran sekali saja dan hanya perlu memanggil fungsi pinggiran (tabel hasil preprocessing) untuk selanjutnya. Misalnya jika terjadi ketidakcocokan pada Pattern[4], maka program langsung tahu bahwa pattern perlu digeser sebanyak 5-1=5 karakter.

III. DASAR HTML

HyperText Markup Language adalah sebuah bahasa untuk membuat halaman web atau aplikasi berbasis web. HTML digunakan untuk mendeskripsikan suatu struktur untuk sebuah web menggunakan *markup*. Sebuah halaman web HTML dibangun dari elemen-elemen HTML yang disebut dengan tags. Tags adalah jenis elemen yang diapit dengan "<>", terdapat banyak tags untuk mendefinisikan suatu konten dari sebuah halaman web seperti heading (<h1>), paragraph ( ), judul halaman (<title>) dan lainnya. Tags digunakan secara berpasangan, yaitu start tag dan end tag (dan) dengan isi konten yang diinginkan dituliskan diantara kedua tag tersebut. Tags tersebut hanya sebagai penanda jenis elemen dari konten yang akan ditampilkan pada halaman web, sehingga tags tersebut tidak akan dituliskan pada halaman web hasil rendernya.

Dalam web scraping, seseorang akan bermain dengan tag HTML untuk memperoleh informasinya. Oleh karena itu, diperlukan pemahaman dasar terhadap tag HTML untuk melakukan web scraping. Format sederhana dari sebuah HTML adalah sebagai berikut:

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Gambar 1. Contoh Format HTML

Sumber: www.w3schools.com

Tags <!DOCTYPE html> digunakan untuk menandai bahwa file HTML tersebut menggunakan HTML5, tag ini tidak perlu diberi endtag. Tag <html> adalah elemen yang wajib untuk sebuah file HTML. <head> adalah tag container untuk elemen head seperti title, scripts, styles, dan meta information yang

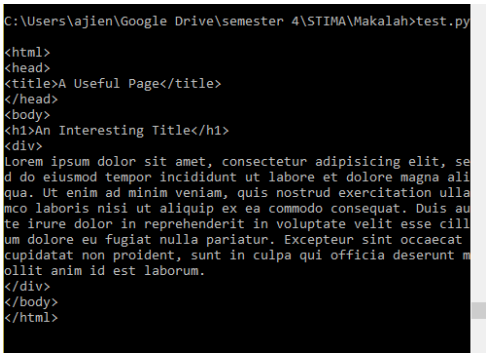
bukan bagian dari konten yang ditampilkan sebuah halaman. Tag <body> adalah tag container untuk elemen-elemen body yang merupakan konten dari sebuah halaman seperti paragraph, gambar, tabel, dan lainnya. Selain itu, beberapa tag yang sering digunakan dalam sebuah file HTML adalah sebagai berikut:

Tag	Deskirpsi
<title>	Judul dokumen
 	Line Break
<hr>	Mendefinsikan perubahan tematik pada konten (Defaultnya diberi line break dengan garis)
<h1> - <h6>	Heading sesuai levelnya, h1 berarti heading paling besar.
<p>	Paragraph
<!--…-->	Komentar
<u>,<i>, <b>, <s>	Underline, italic, bold, dan Strikethrough (coret)
<img>	Gambar
<a>	Hyperlink, yaitu text yang jika di klik akan mengarahkan ke suatu link
<table>	Tabel
dan	Membuat container atau section dari elemen-elemen lainnya. Div untuk satu blok, span untuk inline. Biasanya digunakan untuk styling (advanced), namun penting untuk data scraping.
<style>	Memberikan informasi styling untuk sebuah dokumen HTML
<form>	Formulir untuk input pengguna
<button>	Tombol

IV. Web Scraping

Web scraping atau data scraping adalah suatu teknik pada perangkat lunak komputer untuk mengekstrak suatu informasi atau data dari website. Proses web scraping biasanya dilakukan dengan cara mentransformasi data tidak terstruktur dalam format HTML kedalam bentuk terstruktur seperti database.

Data scraping dapat dilakukan di berbagai bahasa pemrograman seperti java dan python. Banyak bahasa pemrograman juga sudah memiliki library untuk melakukan data scraping dengan mudah seperti beautifulsoup pada python dan JSoup untuk java. Namun tanpa library-library tersebut, data scraping masih dapat dilakukan. Hal yang penting dari sebuah bahasa program agar dapat melakukan data scraping adalah agar dapat melakukan "get request" dari sebuah web. Pada bahasa python, aksi tersebut dapat dilakukan menggunakan fungsi requests.get([link website]). Fungsi ini akan mengembalikan data HTML dari link yang diberikan yang selanjutnya akan diproses untuk diekstrak informasi yang diinginkan.



Gambar 2. Hasil Get dari link http://pythonscraping.com/pages/page1.html

Dari HTML yang didapat dari perintah request diatas dapat diambil beberapa informasi seperti judul yaitu "A Useful Page", dengan salah satu konten dengan penulisan header 1 "An Interesting Title", dan juga isi container (div) yaitu lorem ipsum. Untuk website yang lebih komples, HTML yang didapat juga akan lebih kompleks. Oleh karena itu, diperlukan algoritma untuk mencari data yang diinginkan menggunakan algoritma string matching.

V. Implementasi

Pada data scraping, informasi yang dicari biasanya dilihat dari tag dari kontennya. Misalnya jika ingin mengambil informasi paragraph, maka seseorang hanya perlu mencari tag <p> dan membaca konten dari tag tersebut. Pencarian tag ini dapat dimudahkan dengan membuat fungsi pencarian, misalnya pada implementasi ini menggunakan algoritma KMP. Berikut pseudo-code dari fungsi tersebut.

Dengan html yang dihasilkan dari gambar 2, berikut adalah hasil implementasi findTag dengan mencari tag div dan h1, yaitu findTag('div', html.text) dan findTag('h1', html.text).

```
Function kmp (text:string, pattern:string): integer
Function findTag (tag: string, html:string): string
Kamus
startTag, endTag: String
startIdx, endIdx: Integer
Algoritma
startTag <-  "<" + tag + ">"
endTag <-  "</" + tag + ">"
startIdx <-  kmp(html, startTag)
endIdx <-  kmp(html, endTag)
➔  html[startIdx..endIdx]
```

```
C:\Users\ajien\Google Drive\semester 4\STIMA\Makalah>test.py
<div>
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusm
d tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex e
 commodo consequat. Duis aute irure dolor in reprehenderit in voluptat
 velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occ
ecat cupidatat non proident, sunt in culpa qui officia deserunt mollit
anim id est laborum.
</div>
```

```
C:\Users\ajien\Google Drive\semester 4\STIMA\Makalah>test.py
<h1>An Interesting Title</h1>
```

Gambar 3. Hasil Implementasi findTag

Link yang diberikan pada contoh diatas merupakan halaman yang sangat sederhana. Pada halaman yang lebih kompleks, biasanya tags memiliki beberapa atribut seperti class, id, atau atribut lain yang dapat didefinisikan sendiri oleh developer. Misalkan diambil halaman lain yang sedikit lebih kompleks, yaitu www.pythonscraping.com/pages/warandpeace.html. Halaman tersebut berisikan sebuah cuplikan cerita berjudul War and Peace. Pada halaman tersebut, semua nama tokoh pada cerita tersebut diberi warna hijau dan semua dialog diberi warna merah. Jika ditelusuri HTML-nya, dapat diketahui warna tulisan yang berbeda tersebut menggunakan tags span dengan class sesuai warnanya.

It was in July, 1805, and the speaker was the well-known Anna Pavlovna Scherer, maid of honor and favorite of the Empress Marya Fedorovna. With these words she greeted Prince Vasili Kuragin, a man of high rank and importance, who was the first to arrive at her reception. Anna Pavlovna had had a cough for some days. She was, as she said, suffering from la grippe; grippe being then a new word in St. Petersburg, used only by the elite.

All her invitations without exception, written in French, and delivered by a scarlet-liveried footman that morning, ran as follows:

"If you have nothing better to do, Count [or Prince], and if the prospect of spending an evening with a poor invalid is not too terrible, I shall be very charmed to see you tonight between 7 and 10- Annette Scherer."

Gambar 4. Cuplikan halaman cerita War and Peace

```
<p>
  It was in July, 1805, and the speaker was the well-known
  <span class="green">Anna Pavlovna Scherer</span>
  , maid of honor and favorite of the
  <span class="green">Empress Marya Fedorovna</span>
  . With these words she greeted
  <span class="green">Prince Vasili Kuragin</span>
  , a man of high rank and importance, who was the first to arrive at her reception.
  <span class="green">Anna Pavlovna</span>
  had had a cough for some days. She was, as she said, suffering from la grippe; grippe being then a new word in
  <span class="green">St. Petersburg</span>
  , used only by the elite.
</p>
<p>…</p>
<p>
  "
  <span class="red">
    If you have nothing better to do, Count [or Prince], and if the prospect of spending an evening with a poor invalid is no
    terrible, I shall be very charmed to see you tonight between 7 and 10- Annette Scherer.
  </span>
  "
</p>
```

Gambar 5. File HTML dari Cuplikan Pada Gambar 4.

Oleh karena itu, diperlukan tambahan pada fungsi findTag agar dapat mendukung pencarian Tag beserta atributnya.

Function findTag2 (tag: string, atr: dictionary, html:string): string

//Contoh isi atr : {'class' : 'red'}

- Cari string dari html yang mengandung Tag tag menggunakan KMP seperti fungsi findTag sebelumnya. Misalnya hasil string yang ditemukan adalah S.
- Evaluasi apakah string yang ditemukan memenuhi syarat atribut yang dicari :
    - Untuk pasangan key-value pada atr, cek apakah ada string "'[key]' = '[value]'" pada S menggunakan KMP.
    - Jika ada, kembalikan S
    - Jika tidak ada, lanjutkan mencari Tag lainnya menggunakan KMP dengan html yang sudah dipotong dengan S. Jika masih ada, lakukan evaluasi kembali.

Hasil findTag2('span', {'class':'green'}, html):

```
C:\Users\ajien\Google Drive\semester 4\STIMA\Makalah>test.py
<span class="green">Anna
Pavlovna Scherer</span>
```

Gambar 6. Hasil findTag2, mencari Tag span pertama yang memiliki atribut class = green.

Hasil findTag2('span', {'class':'red'}, html):

```
C:\Users\ajien\Google Drive\semester 4\STIMA\Makalah>test.py
<span class="red">Well, Prince, so Genoa and Lucca are now just family
estates of the
Buonapartes. But I warn you, if you don't tell me that this means war,
if you still try to defend the infamies and horrors perpetrated by
that Antichrist- I really believe he is Antichrist- I will have
nothing more to do with you and you are no longer my friend, no longer
my 'faithful slave,' as you call yourself! But how do you do? I see
I have frightened you- sit down and tell me all the news.</span>
```

Gambar 7. Hasil findTag2, mencari Tag span pertama yang memiliki atribut class = red.

Fungsi findTag dan findTag2 masih memiliki kekurangan, yaitu mereka hanya mengembalikan Tag yang pertama kali ditemukan saja. Padahal, mungkin saja konten yang memiliki suatu Tag yang sama ada banyak. Misalnya pada halaman War and Peace, terdapat beberapa nama yang berada di dalam tag Span dengan atribut 'class' berwarna 'green'. Oleh karena itu perlu fungsi lain yang akan mencari seluruh Tag yang sesuai kriteria dan memasukannya kedalam sebuah array atau list.

Misalkan dibuat fungsi findAll dan findAll2 yang menerima parameter yang sama dengan findTag dan findTag2 dan mengembalikan array dari semua string yang memenuhi kriteria masing-masing fungsi. Misalkan digunakan pendekeatan rekursif untuk implementasi findAll dan findAll2. Berikut alur programnya:

Function findAll (tag: string, html:string): list of string

- Cari Tag pertama yang ditemukan menggunakan findTag(tag, html). Misalkan hasilnya adalah S.
- Jika tidak ada (basis), kembalikan list kosong
- Jika S ada (Rekursif)
    - Definisikan list of string Result.
    - Tambahkan S pada Result.
    - Definisikan newHtml, yaitu html yang sudah dipotong dengan S. Misalkan S ditemukan pada html pada [n..m], maka newHtml = html[m..length(html)-1]
    - Lakukan konkat Result dengan findAll(tag, newHtml).
    - Kembalikan Result

Function findAll2 (tag: string, atr: dictionary, html:string): list of string

- Cari Tag pertama yang ditemukan menggunakan findTag2(tag, atr,html). Misalkan hasilnya adalah S.
- Jika tidak ada (basis), kembalikan list kosong
- Jika S ada (Rekursif)
    - Definisikan list of string Result.
    - Tambahkan S pada Result.
    - Definisikan newHtml, yaitu html yang sudah dipotong dengan S. Misalkan S ditemukan pada html pada [n..m], maka newHtml = html[m..length(html)-1]
    - Lakukan konkat Result dengan findAll(tag, newHtml).Kembalikan Result

Hasil findAll2(‘span’, {‘class’ : ‘green’}, html) dengan halaman War and Peace:

```
C:\Users\ajien\Google Drive\semester 4\STIMA\Makalah>test.py
[<span class="green">Anna
Pavlovna Scherer</span>, <span class="green">Empress Marya
Fedorovna</span>, <span class="green">Prince Vasili Kuragin</span>, <sp
an class="green">Anna Pavlovna</span>, <span class="green">St. Petersbu
rg</span>, <span class="green">the prince</span>, <span class="green">A
nna Pavlovna</span>, <span class="green">Anna Pavlovna</span>, <span cl
ass="green">the prince</span>, <span class="green">the prince</span>, <
span class="green">the prince</span>, <span class="green">Prince Vasili
</span>, <span class="green">Anna Pavlovna</span>, <span class="green">
Anna Pavlovna</span>, <span class="green">the prince</span>, <span clas
s="green">Wintzingerode</span>, <span class="green">King of Prussia</sp
an>, <span class="green">le Vicomte de Mortemart</span>, <span class="g
reen">Montmorencys</span>, <span class="green">Rohans</span>, <span cla
ss="green">Abbe Morio</span>, <span class="green">the Emperor</span>, <
span class="green">the prince</span>, <span class="green">Prince Vasili
</span>, <span class="green">Dowager Empress Marya Fedorovna</span>, <s
pan class="green">the baron</span>, <span class="green">Anna Pavlovna</
span>, <span class="green">the Empress</span>, <span class="green">the
Empress</span>, <span class="green">Anna Pavlovna's</span>, <span class
="green">Her Majesty</span>, <span class="green">Baron
Funke</span>, <span class="green">The prince</span>, <span class="green
">Anna
Pavlovna</span>, <span class="green">the Empress</span>, <span class="g
reen">The prince</span>, <span class="green">Anatole</span>, <span clas
s="green">the prince</span>, <span class="green">The prince</span>, <sp
an class="green">Anna
Pavlovna</span>, <span class="green">Anna Pavlovna</span>]
```

Gambar 8. Hasil findAll2, mencari semua Tag span dengan class=green

Dengan fungsi findTag dan findAll yang sudah dimplementasikan, web scraping akan jauh lebih mudah dan cepat dibandingkan mencari data dari tag secara manual. Fungsi findTag dan findAll akan selalu mengembalikan tag yang dicari hingga end tag-nya, termasuk semua konten didalamnya. Jadi hasil dari keempat fungsi tersebut mungkin menghasilkan nested Tag.

Untuk mendapatkan sebuah Tag yang di-*nested* dari Tag lainnya, dapat digunakan sebuah trik yaitu memanggil fungsi findTag/findAll dua kali. Pertama untuk mencari Tag parent-nya dan yang kedua untuk mencari Tag anaknya. Misalkan dari halaman War and Peace ingin dicari sebuah tag yang di-nest pada tag <p>. Maka dapat dilakukan :

```
findTag(‘span’, findTag(‘p’, html))
//atau untuk mencari semua tagnya:
temp = findAll(‘p’, html)
for tag in temp:
    print(findAll(‘span’, tag))
```

VI. Kesimpulan dan Saran

Algoritma string matching dapat digunakan untuk mempermudah/mempercepat proses web scraping secara manual (tanpa API atau library eksternal). Pada implementasi pada makalah ini algoritma string matching yang digunakan adalah KMP, mungkin untuk kedepannya dapat dibuat perbandingan performa menggunakan algoritma string matching yang lain. Algoritma string matching digunakan untuk mencari sebuah/semua tag yang dinginkan dan berhasil diimplementasikan kedalam bentuk program python. Walau kemampuan fungsi/program yang diimplementasikan pada makalah ini hanya dapat mencari tag, namun penggunaannya dalam web scraping sudah cukup untuk mengambil informasi sederhana dan semi-kompleks, ditambah lagi fungsi-fungsi yang diimplementasikan pada makalah ini dapat digunakan secara kreatif agar mendapatkan informasi lain yang lebih kompleks (seperti mencari nested Tag). Jadi, penggunaan algoritma string matching dapat digunakan untuk membantu proses web scraping dan masih dapat dikembangkan lebih lanjut lagi, misalnya mengambil informasi Atribut dari sebuah Tag (bukan konten dari Tag), atau mencari Tag dimana ditemukan sebuah konten tertentu.

Ucapan Terima Kasih

Pertama saya mengucapkan terima kasih kepada Tuhan Yang Maha Esa yang telah melimpahkan nikmat, ilmu, dan kekuatan sehingga makalah Strategi Algoritma ini dapat terselesaikan tepat pada waktunya dan sesuai dengan ketentuan. Ucapan terima kasih juga penulis sampaikan kepada kedua orang tua yang telah mendukung saya sepenuh hati, baik secara lisan maupun doa, baik dalam pembuatan makalah ini maupun dalam kegiatan perkuliahan secara umum. Terima kasih juga kepada teman-teman dan kakak tingkat yang baik secara langsung maupun tidak membantu saya dalam pembuatan

makalah ini. Tidak terlupakan juga penulis mengucapkan teirma kasih kepada dosen pengajar Strategi Algoritma K02, Ibu Ulfa,

beserta dosen Strategi Algoritma lainnya, Bapak Rinaldi dan Ibu Masayu. Akhir kata, penulis meminta maaf sebesar-besarnya jika terdapat kekurangan pada makalah ini. Penulis berharap makalah ini dapat bermanfaat dan bisa terus dikembangkan untuk banyak orang kedepannya.

REFERENSI

[1] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Pencocokan-String-(2018).pdf, diakses pada 12 Mei 2018.

[2] https://www.w3schools.com, diakes pada 12 Mei 2018.

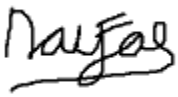
[3] https://www.analyticsvidhya.com/blog/2015/10/beginner-guide-web-scraping-beautiful-soup-python/, diakses pada 12 Mei 2018.

[4] https://github.com/REMitchell/python-scraping, diakses pada 12 Mei 2018.

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, 13 Mei 2018



Naufal Putra Pamungkas
13516110