Transliterasi dan Pencarian Ortografi Arab dalam Alquran Menggunakan Regex

Ilham Syahid Syamsudin - 13518028[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
[1]Ilhamsyahids@gmail.com

***Abstract*— Alquran merupakan kitab suci terakhir sekaligus penutup dan penyempurna kitab sebelumnya yang diturunkan melalui Nabi Muhammad ﷺ yang mana seorang muslim wajib mengimani serta meyakininya. Alquran terdiri atas 114 surah yang didalamnya terkumpul sejumlah ayat. Setiap ayat mengandung beberapa kata (atau kalimat) dan terkandung di dalamnya beberapa huruf. Setiap huruf dapat mengandung sejumlah arti dan jika digabungkan huruf-huruf tersebut, akan mengandung arti yang lainnya. Tulisan ortografi Arab modern termasuk ke dalam karakter spesial. Sehingga dalam penulisannya perlu karakter khusus yang dalam hal ini terdapat pada Unicode. Selain itu, dalam penulisan kata perkatanya dimulai dari kanan yang terkadang sulit bersambung dengan huruf yang dimulai dari kiri. Pencarian dan transliterasi string pada ortografi Arab merupakan pemanfaatan dari algoritma pencarian string. Dalam pembuatan tulisan ini, penulis memanfaatkan library JQuranTree yaitu Java API untuk mengakses teks Alquran dan Regex untuk pencarian string.**

Keywords— Ortografi, Arab, String, Regex, Buckwalter .

I. Introduction

Alquran pertama kali dibukukan pada masa Abu Bakar Ash-Shiddiq رضي الله عنه selaku khalifah pertama, akan tetapi tidak disatukan dalam satu logat. Kemudian pada masa khalifah Utsman bin Affan رضي الله عنه disatukan dalam satu logat yang hingga saat ini dipakai banyak orang. Sebelum hal itu, Alquran diturunkan dengan tujuh logat bahasa Arab dan tidaklah dibukukan akan tetapi dihafalkan [1]. Hingga saat ini, Alquran terus-menerus dihafalkan dan tidak pernah berubah satu kata maupun satu huruf pun.

Sesaat ditemukannya komputer atau alat yang memanfaatkan teks didalamnya, tidaklah terdapat huruf yang bisa merepresentasikan karakter-karakter khusus ini. Seiring berjalannya waktu, teknologi terus berkembang hingga dibuatlah standar ekspresi penulisan teks yang dikenal sebagai Unicode. Unicode Standard merupakan kumpulan dari serangkaian kode untuk merepresentasikan visual, pengkodean karakter, sekaligus tampilan teks yang mengandung *script* kanan ke kiri seperti bahasa Arab dan *script* kiri ke kanan.

Unicode mengandung semua *script* sistem penulisan yang dipakai hingga saat ini. Tidak hanya mengenai bahasa, berbagai karakter rumus, seperti rumus fisika, dapat divisualisasikan oleh standar Unicode.

Terdapat sebuah organisasi non-profit yang bertujuan untuk mengelola dan menerbitkan Unicode Standard bernama Unicode Consortium (Unicode Inc.). Organisasi ini didirikan 3 Januari 1991 yang bertujuan untuk bisa membuat semua orang di seluruh dunia bisa menggunakan komputer dengan bahasa apa saja.

Gambar 1 Logo Unicode Consortium

Selain dari Unicode terdapat definisi lain yaitu ortografi (*orthography*). Ortografi adalah konvensi dari sistem penulisan yang mengandung penulisan khusus.

Terlepas dari bahasa yang digunakan, pencarian string merupakan salah satu hal yang paling berguna dalam kehidupan kita saat ini. Bayangkan jika kita harus mencari sejumlah kata di beberapa dokumen tanpa adanya otomasi, sangat dimungkinkan terjadi akan tetapi membutuhkan waktu yang lama. Saat ini, tidak satupun komputer yang tidak memiliki sistem pencarian string.

Salah satu cara populer untuk mencari string adalah dengan menggunakan Regex. Regex atau *regular expression* adalah

deretan karakter khusus yang mendefinisikan suatu pola tertentu dalam pencarian atau pencocokan string. Teknik ini dapat dengan mudah mencari pola tertentu pada sebuah string atau kumpulan karakter.

Bahasa Arab saat ini dilengkapi oleh tanda titik untuk menandakan perbedaan tiap huruf (*hijaiyah*). Sehingga kita dapat membedakan *ta* (ت) dengan *ba* (ب) dan *tsa* (ث) dan huruf lainnya yang memiliki titik. Selain itu, bahasa Arab juga memudahkan bagi seorang yang baru belajar atau belum mengetahui tata bahasa (*i'rab*) dengan benar dapat dilengkapi dengan tanda vokal (harakat) atau disebut dengan diakritik.

Perlu diketahui bahwa Unicode dapat saling menempati tempat yang sama. Seperti تُ (*tu*) yang memiliki dua Unicode yaitu ت (*ta*) dan ُ(*dammah*). Masing-masing karakter memiliki kode tersendiri. Sehingga pada penulisan bahasa Arab, apabila kita ingin menuliskan dengan bentuk berharakat maka kita harus menuliskan minimal dua Unicode. Sehingga akan sulit jika kita mencari kata dalam bahasa Arab yang tidak memiliki harakat. Seperti kita ingin mencari kata بسم (*bsm*) maka tidak akan kita temui, padahal yang kita maksud adalah mencari kata بِسْمِ (*bismi*, dengan nama).

Dalam kasus tersebut kita dapati dua opsi untuk melakukannya yaitu menghapus harakat terlebih dahulu atau melakukan *string matching* yang dalam hal ini kita menggunakan regex. Opsi pertama bisa kita gunakan jika memang yang kita mencari kata tanpa harakat, tidak berlaku jika ada salah satu pola memakai harakat. Berbeda dengan menggunakan regex, kita dapat memperolehnya dengan atau tanpa harakat. Seperti pada kasus sebelumnya, بسم dapat kita ubah menjadi ب.؟س.؟م.؟ , karakter "." merupakan karakter khusus di regex yang menandakan satu karakter apapun dapat masuk dalam hal ini kita mengharapkan sebuah harakat dan karakter "?" menandakan karakter sebelumnya tidak muncul atau muncul satu kali. Dalam kata lain, karakter tersebut merupakan karakter opsional.

Apabila kita mencari dengan bentuk regex seperti itu, diharapkan kita dapat mencari *kalimah* yang dengan atau tidak memiliki harakat. Pencarian ini tidak terbatas pada teks ortografi Arab, akan tetapi juga pada transliterasi Buckwalter .

Sebagai informasi tambahan kata dalam bahasa Arab disebut *al-kalimah* (الكلمة) dan kalimat dalam bahasa Arab disebut *al-jumlah* (الجملة).

II. Teori Dasar

A. *String Matching*

Algoritma *string matching* atau pencocokan string merupakan algoritma pencarian sebuah pola tertentu dalam sebuah sekumpulan karakter (string). *String matching* merupakan fundamental dalam pemprosesan teks dan dasar dari implementasi pada *software* di sistem operasi. Algoritma string matching banyak kita temui di berbagai lini kehidupan kita saat ini. Seperti misalnya kita mencari sebuah teks di sebuah dokumen atau saat kita mencari di search engine google kita dapati algoritma tersebut. Tidak sebatas pada teks, kita bisa dapati algoritma ini pada pencarian citra atau gambar pada pemanfaatan sidik jari atau kita dapat mencari rantai asam amino pada rantai DNA.

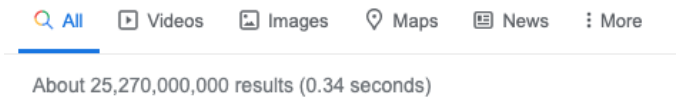
Algoritma ini dibagi menjadi beberapa klasifikasi, algoritma pencarian pola tunggal (*single-pattern*), pencarian pola terbatas (*finite-set-pattern*), pencarian pola tidak terbatas (*infinite-set-pattern*) dan klasifikasi lainnya. *Knuth–Morris–Pratt algorithm* dan *Boyer–Moore string-search algorithm* dapat kita masukkan dalam pencarian pola tunggal dan pencarian pola terbatas. Karena terdapat perluasan dari algoritma tersebut yang masuk kepada pencarian pola terbatas salah satunya adalah *Commentz-Walter algorithm* yang merupakan perluasan dari *Boyer–Moore algorithm*.

Algoritma pencarian lainnya yaitu *naïve-search* yaitu pencarian dengan teknik *brute-force*. Selain itu, terdapat algortima pencarian *finite-state-automaton* yaitu pencarian yang didasari oleh mesin *deterministic finite automaton* (DFA) yang dapat menerima atau menolak sebuah string. Gambar dibawah ini merupakan contoh status-status (state) pada mesin DFA.

Telah kita singgung sebelumnya bahwa algoritma pencarian menjadi dasar dan fundamental di lingkup ilmu komputer. Kenapa algoritma pencarian ini sangatlah penting? Kenapa kita harus memilih algoritma yang tepat?

Bayangkan Google yang memiliki pencarian tiap harinya. Kita menyadari bahwa kita tidak ingin menunggu banyak waktu, kita ingin response yang cepat yang diberikan oleh Google. Ratusan bahkan ribuan data yang perlu dicari untuk mendapatkan apa yang kita inginkan. Ini merupakan tugas dari algoritma pencarian. Oleh karena, pemilihan algoritma pencarian sangatlah penting.

Sebagai contoh, kita memilih algoritma yang membutuhkan waktu satu detik untuk setiap hasilnya.



Maka perlu berapa lama untuk mendapatkan hasil diatas? Jawabannya 25.270.000.000 detik atau 800 tahun! Tentu kita tidak ingin menghabiskan 800 tahun untuk satu kali pencarian. Pencarian dengan algoritma yang tepat dan benar memberikan kita aplikasi yang baik.

Pada dasarnya, string S merupakan sekumpulan karakter (*array of character*) dengan panjang m, maka dapat kita asumsikan bahwa $S = x_0x_1x_2\ldots x_{m-1}$. Prefix dari S adalah sebuah substring S[0…k] dan suffix dari S merupakan sebuah substring S[k…m-1] dengan k adalah index diantara 0 dan m-1.

Jika terdapat string “Ilham” maka prefix dari S adalah “I”, “Il”, “Ilh”, “Ilha”, “Ilham” dan suffix dari S adalah “m”, “am”, “ham”, “lham”, “Ilham”.

B. *Regular Expression (Regex)*

Seperti yang telah disinggung *Regular Expression* atau Regex merupakan deretan karakter khusus yang mendefinisikan suatu pola tertentu dalam pencarian atau pencocokan string. Secara formal, *regular expression* adalah cara aljebra mengekspresikan *languages*. Kita dapat mendefinisikannya secara rekursif [5].

Misal terdapat *regular language* dari ∑ maka:

1. Bahasa kosong Ø dan bahasa yang mengandung satu string yang panjangnya satu merupakan *regular expression*
2. Jika terdapat simbol e, dengan e termasuk Σ maka L(e)={e} merupakan *regular expression*. Notasi {e} merupakan bahasa yang mengandung satu string dengan panjang satu.
3. Jika E1 and E2 adalah *regular expressions*, maka E1+E2 adalah *regular expressions*, dan L(E1+E2) = L(E1) ∪ L(E2).
4. Jika E1 and E2 adalah regular expressions, maka E1•E2 adalah regular expression, dan L(E1E2) = L(E1)L(E2). L(E1)•L(E2) merupakan konkatenasi yang berarti string wx ialah w pada L(E1) dan x pada L(E2).
5. Jika E adalah *regular expression*, maka E* adalah regular expressions, juga L(E*) = (L(E))*. E* disebut Kneele Star dari E yaitu E0 ∪ E1 ∪ E2 ∪ …

Dengan kata lain, *regular expression* merupakan kelanjutan dari teori *finite automata* yang dapat dibaca lebih lanjut di literatur lainnya.

C. *Struktur Alquran*

Alquran terdiri dari 114 surah (سوره) dimulai dari Al-Fatihah sampai An-Nas dan mengandung 6236 ayat (آيات), dalam *rasm* Utsmani. Dalam implementasi ini, penulis menggunakan library JQuranTree yaitu sebuah library yang mengimplementasikan Alquran kedalam struktur data yang lebih mudah ditelusuri oleh program. Alquran yang digunakan ialah Alquran dengan *rasm* Utsmani yang dapat secara bebas di unduh di http://tanzil.net/download. Teks Utsmani merupakan Unicode dari Mushaf Madinah.

Secara umum, *Arabic orthographic* dapat kita modelkan sebagai berikut:

Gambar 2 Model Objek Ortografi [4]

Model ortografi Alquran ini bersifat *immutable*, yang berarti model tersebut dapat dibaca dan dicari akan tetapi tidak dapat di ubah. Berikut ini adalah definisi dari model ortografi diatas:

1. Document

Struktur yang merepresentasikan keseluruhan teks Alquran termasuk di dalamnya surah, ayat dan informasi tambahan lainnya.

2. Chapter (Surah)

Surah dalam Alquran yang berjumlah 114. Surah memiliki nama dan nomor yang unik. Setiap surah diawali dengan bismillah kecuali surah At-Taubah.

3. Verse (Ayat)

Setiap surah dibagi menjadi beberapa ayat. Dalam rasm Utsmani, terdapat 6236 ayat.

4. Token

Ortografi yang merepresentasikan tiap *al-kalimah* (kata) tanpa yang sudah dipisahkan dengan *whitespace* didalam tiap ayatnya. Di dalam bahasa Arab, terkadang kata dasar dengan imbuhannya tergabung dalam satu ortografi.

5. Character

Bentuk ini memodelkan suatu huruf (*hijaiyah*) dalam bahasa Arab.

6. Diacritic

Model ini bisa kita sebelum dengan huruf vokal atau secara umum disebut harakat.

Seperti yang telah disinggung sebelumnya, teks Arab pada Alquran berbeda dengan teks Arab modern yang dipakai sehari-hari. Teks modern jarang bervokal (tidak ada harakat), sedangkan Alquran berisi serangkaian harakat yang komprehensif. Diakritik Arab sendiri pada awalnya diperkenalkan untuk mempermudah aksara Arab dalam Alquran. Tanda ortografis umumnya ditemukan dalam Alquran yang jarang ditemukan dalam teks-teks lain. Tanda-tanda ini dan lainnya dalam Alquran digunakan untuk mendukung keakuratan pengucapan ataupun keakuratan makna[4].

Gambar 3 Bentuk Penulisan Bahasa Arab

Penjelasan tiap bentuknya sebagai berikut:

1. Bentuk formal umum karakter.
2. Bentuk bertitik.
3. Bentuk bervokal (harakat).
4. Bentuk penulisan pembacaan.

D. Transliterasi

Transliterasi merupakan bentuk penulisan lambang khusus kedalam bentuk lain yang memiliki bunyi yang sama atau mendekatinya. Seperti contoh pada kata بِسْمِ , kita menuliskannya dengan *bismi* karena memiliki bunyi yang sama. Contoh lain ialah ∑ kita bisa menyebutnya *sigma*. Salah satu penulisan transliterasi bahasa Arab ialah Buckwalter . Transliterasi Buckwalter dikembangkan pada ALPNET Arabic Project yang dijalankan oleh Dr. Ken Beeley pada 1988 [6].

Hampir semua bentuk dalam bahasa Arab ditransliterasikan ke dalam karakter khusus. Sebagai contoh pada transliterasinya ialah bentuk بَعْضًا diubah menjadi baEoDFA, yakni ب, ◌َ, ع, ◌ْ, ض, ◌ً, ا masing-masing diubah menjadi b, a, E, o, D, F, A. Karakter lainnya dapat kita lihat pada tabel Buckwalter [6].

III. IMPLEMENTASI

Ada beberapa yang akan kita ujikan pada implementasi ini, yaitu:

1. Latin tanpa harakat
2. Buckwalter dengan harakat
3. Buckwalter tanpa harakat
4. Arabic dengan harakat
5. Arabic tanpa harakat

Latin yang dimaksud adalah teks transliterasi yang biasa diucapkan oleh orang indonesia ketika pertama kali mendengar kata bahasa Arab. Seperti ز yang saat kita mendengarnya berpikir bahwa terdapat huruf z didalamnya.

Pada poin (1), karena tidak didapati acuan transliterasi untuk bahasa Indonesia yang tanpa karakter khusus (keyboard biasa), maka dibuat daftar yang merepresentasikan kata bahasa Indonesia – Buckwalter – Arab. Berikut ini cuplikan daftar yang dibuat:

```
...
sh[a|i|u]? S ص
to[a|i|u]? T ط
tho[a|i|u]? Z ظ
ain[a|i|u]? E ع
gh[a|i|u|o]? g غ
f[a|i|u]? f ف
...
```

Perhatikan bahwa untuk representasi bahasa Indonesia, dibuat dalam bentuk regex karena pada ortografi Arab semisal ف dapat kita tulis dengan *fa*, *fi*, *fu*, atau bahkan hanya *f*.

Sistem encoding yang dibuat adalah kita akan mencoba menggantikan setiap huruf pada input pertama (latin) menjadi bentuk ortografi Arab dan bentuk Buckwalter. Oleh karena itu, daftar tersebut kita iterasi hingga semua huruf latin tergantikan.

Kita akan mencoba menguji pada kasus pertama, yaitu huruf latin di-transliterasikan lalu kemudian dicari kata yang tepat pada Alquran.

```
Input
1. Latin (without harakat)
2. Bulkwalter with harakat
3. Bulkwalter without harakat
4. Arabic with harakat
5. Arabic without harakat
1
bismi llh
Input: bismi llh
Bulkwalter: bsm Allh
Arab: بسم الله
ChapterNumber VerseNumber Verse
------------- ------------ -----
1             1            bisomi {ll~ahi {lr~aHoma`ni {lr~aH
11            41           waqaAla {rokabuwA@ fiyhaA bisomi {
27            30           <in~ahu, min sulayoma`na wa<in~ahu

Matches: 3
(1:1):
بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ
(11:41):
وَقَالَ ٱرْكَبُوا۟ فِيهَا بِسْمِ ٱللَّهِ مَجْرٖىٰهَا وَمُرْسَىٰهَآ إِنَّ رَبِّى لَغَفُورٌ رَّحِيمٌ
(27:30):
إِنَّهُۥ مِن سُلَيْمَٰنَ وَإِنَّهُۥ بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ
```

Gambar 4 Pengujian Pertama, Huruf Latin

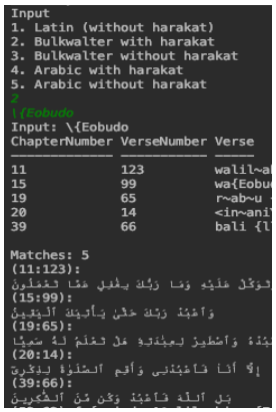
Pada pengujian input latin ini, kita mencoba memasukkan kata bismillah, akan tetapi perlu diketahui bahwa kata tersebut tersusun dari dua *al-kalimah* yang artinya "Dengan (menyebut) nama Allah" sehingga menjadi *bismi llh*.

Langkah selanjutnya ialah mengubah bentuk latin tersebut menjadi bentuk Buckwalter, bentuk yang seharusnya pada potongan kata *bismi* atau بِسْمِ ialah *bisomi* akan tetapi pada implementasi ini, harakat yang menyatu pada huruf konsonan

dijadikan satu sehingga menjadi *bsm*. Bentuk "ll" dalam potongan kata tersebut sejatinya ialah berbentuk الل sehingga jika dijadikan bentuk Buckwalter menjadi *All.*

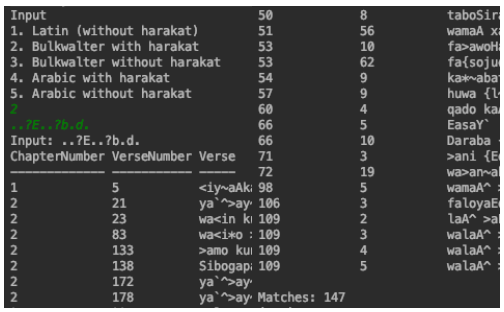
Untuk mengubah menjadi bentuk bahasa Arab, kita dapat memperolehnya dari bentuk Buckwalter yang mengacu pada daftar transliterasi yang sudah dibuat sebelumnya. Sehingga dari *bsm Allh* menjadi بسم الله .

Pada pengujian kedua, kita ambil potongan *al-kalimat* yaitu ٱعۡبُدۡ yang artinya "sembahlah". Jika kita ubah menjadi bentuk Buckwalter , menjadi \{Eobudo. Berikut hasilnya:



Gambar 5 Pengujian Kedua, Buckwalter Berharakat

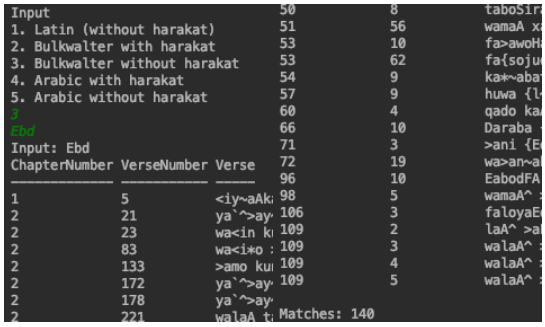
Perlu diperhatikan bahwa "{" merupakan karakter khusus di regex, sehingga perlu karakter escape "\". Dalam pencarian kata ٱعۡبُدۡ kita dapati lima hasil yang sama. Akan tetapi kita ingin semua hasil yang memiliki bentuk dasar "sembah" seperti menyembah (أَعۡبُد), penyembah (عَٰبِد) dan yang lainnya. Sehingga dapat kita ubah menjadi *..?E..?b.d.* .



Gambar 6 Pengujian Kedua, Buckwalter dengan Regex

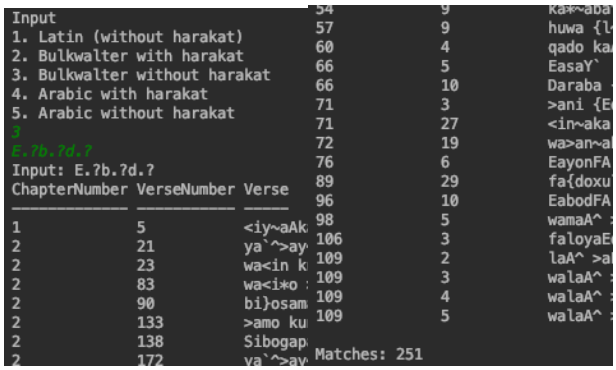
Sehingga pada hasil yang terakhir kita mendapati sebanyak 147 hasil. Tentu hasil tersebut belum semua terambil karena tidak semua bentuk disebutkan pada pola regex sebelumnya.

Pada bentuk yang ketiga, bentuk Buckwalter tanpa harakat. Metode ini akan melakukan pencarian terhadap ayat berbentuk transliterasi Buckwalter yang sebelumnya huruf vokalnya dihapus. Sehingga pada pencarian sebelumnya yang mana kita harus mencarinya dengan harakat, kini tidak memerlukannya. Kita akan mencoba menjadi pencarian sebelumnya dengan *Ebd*.



Gambar 7 Pengujian Ketiga, Buckwalter Tanpa Harakat

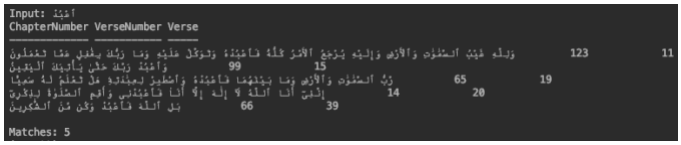
Kita langsung mendapati menjadi 140 hasil. Tentu bentuk ini belum semua terambil dari kata dasarnya. Mari kita coba yang lebih yaitu *E.?b.?d.?* .



Gambar 8 Pengujian Ketiga, Buckwalter dengan Regex

Hasilnya kita dapati menjadi 251 hasil. Hal ini karena bentuk turunan dari kata dasar dalam bahasa Arab melibatkan banyak huruf konsonan tergantung *wazn* (bentuk) dan *i'rab* (*grammar* dalam bahasa Inggris). Lebih lengkap mengenai perubahan bentuknya dapat kita pelajari di dalam ilmu *sharaf*.

Untuk implementasi poin (4) dan (5) tidak akan jauh beda dengan poin sebelumnya, hanya saja yang membedakan adalah menggunakan bahasa Arab. Mari kita coba seperti pada implementasi sebelumnya.



Gambar 9 Pengujian Keempat, Arab Berharakat

Pencarian tersebut sama seperti pada pengujian kedua yaitu menggunakan kata ٱعۡبُدۡ yaitu didapati lima hasil. Disebabkan oleh script dari kanan ke kiri, sehingga susunan pada tabel pun terbalik. Kolom paling kanan seharusnya menjadi kolom paling kiri dan seterusnya.

Pada pengujian terakhir, kita menggunakan uji kasus seperti pada pengujian ketiga yaitu Ebd dan *E.?b.?d.?* . Jika dijadikan ke dalam bahasa Arab menjadi عبد dan د?.ب?.ع.? . Berikut ini adalah hasilnya:

Gambar 10 Pengujian Kelima, Arab Tanpa Harakat

Gambar 11 Pengujian Kelima, Arab dengan Regex

Terlihat bahwa kedua hasil tersebut menunjukkan hasil yang sama seperti pada pengujian sebelumnya.

IV. KESIMPULAN

Setelah kita melihat implementasi tersebut, terlihat bahwa pencocokan string tidak terbatas pada persoalan keputusan yaitu ada atau tidak adanya suatu pola dalam string. Akan tetapi kita juga dapat memanipulasi string tersebut menjadi bentuk lainnya, bahkan kita dapat memperoleh informasi yang berguna. Dalam hal ini kita dapat mencocokan string pada bahasa Arab yang memiliki karakter tak biasa pada keyword QWERTY secara umum. Selain itu, telah ditunjukkan juga pengubahan karakter (transliterasi) latin bahasa Indonesia menjadi bentuk Buckwalter dan bahasa Arab.

Dengan demikian pencocokan string terkhusus dengan menggunakan regex, kita dapat "bermain" atau melakukan hal yang menarik yang dapat kita eksplor menggunakan string.

VIDEO LINK AT YOUTUBE

https://youtu.be/jTb_UmLrpH4

ACKNOWLEDGMENT

Penulis bersyukur kepada Allah yang telah memberikan hidayah serta rahmat-Nya. Kemudian penulis berterima kasih kepada seluruh dosen pengampu mata kuliah "IF2211 Strategi Algoritma" terkhusus kepada Ibu Masayu Leylia Khodra sekalu dosen kelas mata kuliah tersebut. Dalam penyusunan makalah ini, penulis bereksplorasi sekaligus belajar terkait topik terkait. Penulis juga berterima kasih kepada semua orang yang telah membantu penulis dalam penyusunan makalah ini dan semoga pembaca dapat terinspirasi dengan makalah ini.

REFERENCES

[1] Badri, Arifin. "*Pembukuan Al-Quran*" [Online]. https://konsultasisyariah.com/10-pembukuan-al-quran.html (Diakses 1 Mei 2020)

[2] Charras, Christian dan Thierry Lecroq. "*EXACT STRING MATCHING ALGORITHMS*" [Online]. https://www-igm.univ-mlv.fr/~lecroq/string/node2.html#SECTION0020 (Diakses 1 Mei 2020)

[3] Habash, Nizar. *Introduction to Arabic Natural Language Processing*. Morgan & Claypool, 2010.

[4] Dukes, Kais. "*The Quranic Arabic Corpus*" [Online]. http://corpus.quran.com/ (Diakses 1 Mei 2020)

[5] ITB Lecturers. "IF2220 - Teori Bahasa Formal dan Otomata". Bandung Institute of Technology.

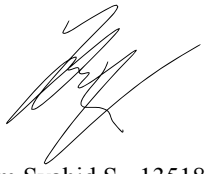
[6] "Buckwalter Transliteration" [Online]. http://www.qamus.org/transliteration.htm (Diakses 1 Mei 2020)

[7] ITB Lecturers. "IF2211 Strategi Algoritma". Bandung Institute of Technology.

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, 1 Mei 2020



Ilham Syahid S - 13518028