PENGEMBANGAN SISTEM PENCARIAN MULTI TEKS PADA TUJUH KITAB HADIS MENGGUNAKAN ALGORITMA *KNUTH-MORRIS-PRATT*

(Skripsi)

Oleh

RAKA NURPANDI

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS LAMPUNG

BANDAR LAMPUNG

2019

ABSTRACT

DEVELOPMENT OF MULTI TEXT SEARCHING SYSTEM IN THE SEVEN BOOKS OF HADITH USING THE KNUTH-MORRIS-PRATT ALGORITHM

By

RAKA NURPANDI

At this moment the roles of technology had cleared away in all areas. It is also in religious area. APJII noted that system searching service treaded on third level at internet user activity in Indonesia Moslems which were user this service often utilized searching services to do excavation of Hadith information. Consequently by developing multitext searching system at seven Hadith books were hope that it could help Moslems in finding information and valid law sourced. This developing method searching system used Knuth-Morris-Pratt Algorithm. This algorithm was one of string matching algorithm for using pattern in the text. This algorithm phase had two steps to determine reshuffle value of pattern whereas searching was step to determine pattern in the text. Besides, this Hadith searching system features to correct spelling mistakes at keywords that entered by users. The totality of Hadith which used was 52.618 Hadith. The main result got at Hadith searching system which developed was system had been able to do searching multi words. System

could display the result of Hadith search on 1 to 5 keywords entered by users. Besides being able to display the results of relevant hadith searches, the system had also been able to correct writing errors on keywords. This searching system needed relevant searching time with amount keywords and searched choise Hadith increasingly was more many keywords and Hadith choise. So system would be more longer to show the searching result.

Keywords: *Hadith*, *String Matching*, *Spell Corrector*, *Knuth-Morris-Pratt Algorithm*.

ABSTRAK

PENGEMBANGAN SISTEM PENCARIAN MULTI TEKS PADA TUJUH KITAB HADIS MENGGUNAKAN ALGORITMA *KNUTH-MORRIS-PRATT*

Oleh

RAKA NURPANDI

Saat ini peran teknologi sudah merambah ke seluruh bidang, tak terkecuali bidang agama. Peran teknologi tersebut selaras dengan pertumbuhan pengguna internet khususnya di Indonesia. APJII mencatat bahwa layanan sistem pencari menduduki peringkat ketiga dalam aktivitas pengguna internet di Indonesia. Umat Muslim yang merupakan pengguna layanan tersebut sering memanfaatkan layanan pencari dalam melakukan penggalian informasi Hadis. Sehingga dengan mengembangkan sistem pencarian multi teks pada tujuh kitab Hadis diharapkan dapat membantu umat Muslim dalam menemukan informasi dan sumber hukum yang valid. Metode dalam pengembangan sistem pencarian ini menggunakan Algortima *Knuth-Morris-Pratt*. Algortima tersebut merupakan salah satu *String Matching Algorithm* untuk menemukan kata kunci *(pattern)* di dalam suatu teks. Fase pencarian algorima ini memiliki dua tahap, yaitu *pre-processing* dan *searching*. *Pre-processing* merupakan tahap untuk menentukan nilai pergeseran suatu *pattern*, sedangkan

searching ialah tahap dalam menemukan *pattern* di dalam suatu teks. Selain itu, sistem pencarian hadis ini memiliki fitur memperbaiki kesalahan ejaan pada kata kunci yang dimasukkan oleh pengguna. Total keseluruhan Hadis yang digunakan adalah 52.618 Hadis. Hasil utama yang didapatkan pada sistem pencarian hadis yang dikembangkan adalah sistem telah mampu melakukan pencarian multi teks. Sistem dapat menampilkan hasil pencarian Hadis pada 1 hingga 5 kata kunci yang dimasukkan pengguna. Selain dapat menampilkan hasil pencarian Hadis yang relevan, sistem juga telah mampu memperbaiki kesalahan penulisan pada kata kunci. Sistem pencarian ini memerlukan waktu pencarian yang selaras dengan jumlah kata kunci dan pilihan hadis yang dicari. Semakin banyak kata kunci dan semakin banyak pilihan hadis, maka sistem semakin lama dalam menampilkan hasil pencarian.

Kata kunci: Hadis, *String Matching*, *Spell Corrector*, Algortima *Knuth-Morris-Pratt*.

PENGEMBANGAN SISTEM PENCARIAN MULTI TEKS PADA TUJUH KITAB HADIS MENGGUNAKAN ALGORITMA *KNUTH-MORRIS-PRATT*

Oleh

RAKA NURPANDI

Skripsi

Sebagai Salah Satu Syarat untuk Mencapai Gelar
SARJANA ILMU KOMPUTER

Pada

Jurusan Ilmu Komputer
Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2019

Judul Skripsi	:	**PENGEMBANGAN SISTEM PENCARIAN MULTI TEKS PADA TUJUH KITAB HADIS MENGGUNAKAN ALGORITMA *KNUTH-MORRIS-PRATT***
Nama Mahasiswa	:	Raka Nurpandi
No. Pokok Mahasiswa	:	1517051016
Jurusan	:	Ilmu Komputer
Fakultas	:	Matematika dan Ilmu Pengetahuan Alam

MENYETUJUI

1. Komisi Pembimbing

Dwi Sakethi, S.Si., M.Kom.
NIP. 19680611 199802 1 001

Ardiansyah, S.Kom., M.Kom.
NIP. 19870128 201803 1 001

2. Mengetahui
Ketua Jurusan Ilmu Komputer
FMIPA Universitas Lampung

Dr. Ir. Kurnia Muludi, M.S.Sc.
NIP. 19640616 198902 1 001

MENGESAHKAN

1. Tim Penguji

   Ketua : **Dwi Sakethi, S.Si., M.Kom.** .....

   Sekretaris : **Ardiansyah, S.Kom., M.Kom.** .....

   Penguji
   Bukan Pembimbing : **Drs. Rd. Irwan Adi Pribadi, M.Kom.** .....

2. a.n Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam
   Wakil Dekan Bidang Akademik dan Kerjasama

Prof. Dr. Sutopo Hadi, S.Si., M.Sc.
NIP 19710415 199512 1 001

Tanggal Lulus Ujian Skripsi : **08 Februari 2019**

PERNYATAAN

Saya yang bertanda tangan di bawah ini, menyatakan bahwa skripsi saya yang berjudul "Pengembangan Sistem Pencarian Multi Teks Pada Tujuh Kitab Hadis Menggunakan Algoritma *Knuth-Morris-Pratt*" merupakan karya saya sendiri dan bukan karya orang lain. Seluruh tulisan yang tertuang di dalam skripsi ini telah mengikuti kaidah penulisan karya ilmiah Universitas Lampung. Apabila di kemudian hari terbukti skripsi saya merupakan hasil penjiplakan atau dibuat oleh orang lain, maka saya bersedia menerima sanksi berupa pencabutan gelar yang telah saya terima.

Bandar Lampung, 11 Februari 2019

RAKA NURPANDI
NPM. 1517051016

RIWAYAT HIDUP

Penulis dilahirkan pada tanggal 15 Desember 1997 di Kotabumi, Lampung Utara. Penulis merupakan anak kedua dari dua bersaudara dengan Ibu bernama Rustiawati dan Ayah bernama Waspan. Penulis menempuh pendidikan formal pertama kali di Raudhatul Athfal Nurul Huda Madukoro pada tahun 2002 hingga 2003. Selanjutnya penulis melanjutkan studi di SD Negeri 01 Madukoro Baru pada tahun 2003 hingga 2009. Pendidikan Sekolah Menengah Pertama di SMP Negeri 6 Kotabumi mulai tahun 2009 hingga 2012. Selanjutnya pada tahun 2012 sampai 2015 penulis menempuh studi di SMA Negeri 2 Kotabumi.

Pada tahun 2015 penulis terdaftar sebagai mahasiswa Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam (FMIPA) Universitas Lampung melalui jalur Seleksi Nasional Masuk Perguruan Tinggi Negeri (SNMPTN). Pada bulan Januari hingga Maret 2018, penulis melakukan kerja praktik di Dinas Komunikasi dan Informatika Kabupaten Lampung Utara selama 40 hari.

Selama menjalani peran mahasiswa, penulis aktif dalam beberapa kegiatan diantaranya:

1. Unit Kegiatan Mahasiswa Fakultas (UKMF) Natural FMIPA Universitas Lampung dengan menjabat sebagai Magang pada tahun 2015, Reporter Media *Online* pada periode 2015-2016, Redaktur Media Dalam Jaringan periode 2016, dan Pemimpin Redaksi pada periode 2017.
2. Peserta Karya Wisata Ilmiah (KWI) di Pekon Batutegi, Kecamatan Air Naningan, Kabupaten Tanggamus pada Januari 2016.
3. Asisten Dosen dan Praktikum beberapa mata kuliah di Jurusan Ilmu Komputer pada periode 2016-2017.
4. Koordinator Divisi Desain, Dekorasi, dan Dokumentasi pada pelaksanaan KWI di Desa Margosari, Kecamatan Pagelaran Utara, Kabupaten Pringsewu pada Januari 2017.
5. Penulis melaksanakan Kuliah Kerja Nyata Kebangsaan di Pekon Sinar Sekampung, Kecamatan Air Naningan, Kabupaten Tanggamus, Provinsi Lampung pada Juli 2018.
6. Penulis juga aktif sebagai Pemateri pada Pelatihan Dasar-Dasar Pemrograman oleh Himpunan Mahasiswa Jurusan Ilmu Komputer tahun 2016, Pelatihan Videografi dan Photoshop oleh Himpunan Mahasiswa Jurusan Biologi tahun 2017, Pelatihan Jurnalistik Tingkat Dasar tahun 2017 oleh UKMF Natural, Pelatihan Desain dan Editing Video oleh Ikatan Mahasiswa Lampung Timur tahun 2018, dan Sekolah Media oleh Badan Eksekutif Mahasiswa (BEM) FMIPA Universitas Lampung tahun 2018. Selain itu, penulis juga menjadi Juri dalam lomba *World Environment Day Article Contest* yang diselenggarakan oleh BEM FMIPA Universitas Lampung tahun 2017.

MOTTO

"Barang siapa yang bersungguh sungguh, sesungguhnya kesungguhan tersebut untuk kebaikan dirinya sendiri"
(Q.S. Al-Ankabut: 6)

"Harga kebaikan manusia adalah diukur menurut apa yang telah dilaksanakan atau diperbuatnya"
(Ali bin Abi Thalib)

"Kegagalan hanya terjadi bila kita menyerah."
(Lessing)

"Tiadanya keyakinanlah yang membuat orang takut menghadapi tantangan; dan saya percaya pada diri saya sendiri."
(Muhammad Ali)

PERSEMBAHAN

Puji dan syukur saya ucapkan kepada Allah SWT atas segala nikmat dan karunia-Nya sehingga skripsi ini dapat diselesaikan.

Kupersembahkan karya ini untuk:

Ibundaku Rustiawati, dan Ayahku Waspan tersayang

Terima kasih atas do'a, kasing sayang, perhatian, dan tetesan keringat yang tak terhitung nilainya.

Kakak tercinta, Esawati Kartika Dewi

Terima kasih telah menjadi teman berantem sejak kecil, dan telah memberikan dukungan terbaik yang tak terlupakan.

Semua Teman Dekatku

Terima kasih telah membersamai, memberikan saran dan motivasi terbaik hingga saat ini.

Keluarga Ilmu Komputer 2015

Keluarga Natural

Universitas Lampung

SANWACANA

Puji sukur penulis ucapkan kehadirat Allah SWT atas berkat rahmat dan hidayah-Nya sehingga skripsi ini dapat diselesaikan. Skripsi dengan judul "Pengembangan Sistem Pencarian Multi Teks Pada Tujuh Kitab Hadis Menggunakan Algoritma *Knuth-Morris-Pratt*" adalah salah satu syarat untuk memperoleh gelar sarjana Ilmu Komputer di Universitas Lampung.

Dalam kesempatan ini penulis mengucapkan terima kasih kepada:

1. Kedua orang tua tercinta, Ibu Rustiawati dan Ayah Waspan, Kakakku tersayang Esawati Kartika Dewi, dan seluruh Keluarga Besar yang selalu memberikan do'a, dukungan, dan semangat kepada penulis.
2. Bapak Dwi Sakethi, S.Si., M.Kom., sebagai pembimbing utama yang telah memberikan ide, saran, dan motivasi kepada penulis.
3. Bapak Ardiansyah, S.Kom., M.Kom., sebagai pembimbing kedua yang telah memberikan ide, saran, dan motivasi kepada penulis.
4. Bapak Drs. Rd. Irwan Adi Pribadi, M.Kom., sebagai pembahas yang telah memberikan saran kepada penulis dalam perbaikan penulisan skripsi.
5. Bapak Prof. Warsito, S.Si., D.E.A., Ph.D., sebagai Dekan FMIPA Universitas Lampung.
6. Bapak Dr. Ir. Kurnia Muludi, M.S.Sc., selaku Ketua Jurusan Ilmu Komputer Universitas Lampung.

7. Bapak Didik Kurniawan, S.Si., M.T., sebagai Sekretaris Jurusan Ilmu Komputer Universitas Lampung.
8. Bapak Febi Eka Febriansyah, S.T., M.T., sebagai Pembimbing Akademik selama penulis menjadi Mahasiswa Ilmu Komputer Universitas Lampung.
9. Bapak dan Ibu Dosen Jurusan Ilmu Komputer yang telah memberikan ilmu selama penulis menjadi mahasiswa.
10. Ibu Ade Nora Maela, Kak Zainuddin, dan Mas Ardi Novalia yang telah membantu segala urusan administrasi di Jurusan Ilmu Komputer.
11. Kak Desy Kartika Sari yang telah memberikan *database* Tujuh Kitab Hadis kepada penulis.
12. Keluarga UKMF Natural FMIPA Universitas Lampung yang telah memberikan semangat kepada penulis.
13. Sahabat sekelik sekaligus rekan kerja, Adji Pangestu dan Fahrul Efendi yang selalu memberikan saran dan motivasi.
14. Teman berbagi keluh kesahku, Yudha, Akmal, Rahmad, Desti, Eliza, Sisil, Sitros, Indri, Asti, Revi, Pipit, dan Ijul.
15. Teman kuliah malam sekaligus teman berpolitikku, Dimas Aji Sukma, Afan Darmaji, dan Fernatdi Angger.
16. Teman sekaligus mentorku dalam menulis, Mba Umi, Mba Nora, dan Kak Shela yang telah membimbingku dengan sabar.
17. Teman seperjuanganku satu kelas “Classic A” yang aku sayangi.
18. Keluarga Besar Ilmu Komputer 2015 yang tak bisa disebutkan satu per satu, terimakasih atas kebersamaannya sejak awal mahasiswa baru.

19. Teman sekaligus adik-adikku, Bella, Indah, Maul, dan Dedek yang selalu memberikan tawa saat bertemu.
20. Keluarga Himakom yang telah menjadi pasangan saat menjalani organisasi kampus.
21. Teman-teman Asisten Dosen yang telah mengajari banyak hal dalam penyampaian ilmu kepada praktikan.
22. Semua pihak yang telah membantu penulis dalam menyelesaikan skripsi ini yang tidak dapat disebutkan satu persatu.
23. Almamater Tercinta.

Penulis menyadari bahwa skripsi ini memiliki kekurangan, sehingga memerlukan saran yang membangun agar menjadi lebih baik. Penulis berharap skripsi ini dapat bermanfaat sebagai sumber informasi dan literatur bagi penulisan dan penelitian karya ilmiah selanjutnya.

Bandar Lampung, 11 Februari 2019
Penulis

Raka Nurpandi

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Gambar Halaman

DAFTAR TABEL

Tabel Halaman

DAFTAR KODE

Kode Halaman

I. PENDAHULUAN

A. Latar Belakang

Revolusi industri saat ini sudah menapaki Era Ke Empat, dimana perangkat digital dapat saling berkaitan satu sama lain. Revolusi ini membawa gaya hidup baru dengan memberikan inovasi besar dalam berbagai bidang kehidupan, seperti super komputer ataupun robot pintar. Dalam bidang teknologi, beragam aplikasi yang tersebar dengan fasilitas internet memberikan kemudahan dalam mengerjakan pekerjaan. Dengan adanya peran teknologi tersebut, waktu, dan jarak seakan dipangkas sehingga manusia dapat melakukan berbagai kegiatan dari jarak jauh (Schwab, 2016).

Perkembangan revolusi tersebut tak lepas dari perkembangan internet dalam menunjang keberadaan teknologi di dalamnya. Berdasarkan data dari Asosiasi Penyelenggara Jasa Internet Indonesia (APJII), pengguna internet di Indonesia pada tahun 2017 mencapai 143,26 juta jiwa dari total populasi penduduk Indonesia sebesar 262 juta jiwa. Selain itu, jika melihat dari pertumbuhan pengguna internet di Indonesia 5 tahun terakhir, total pengguna internet tahun 2013 sebesar 82 juta pengguna. Tahun 2014 sampai 2015 kembali naik hingga 88,1 juta dan 110,2 juta pengguna. Tahun 2016 lalu mencapai 132,7 juta pengguna. Melihat dari data tersebut, kebutuhan internet khususnya di

Indonesia sangatlah besar, salah satunya adalah melakukan pencarian informasi di internet.

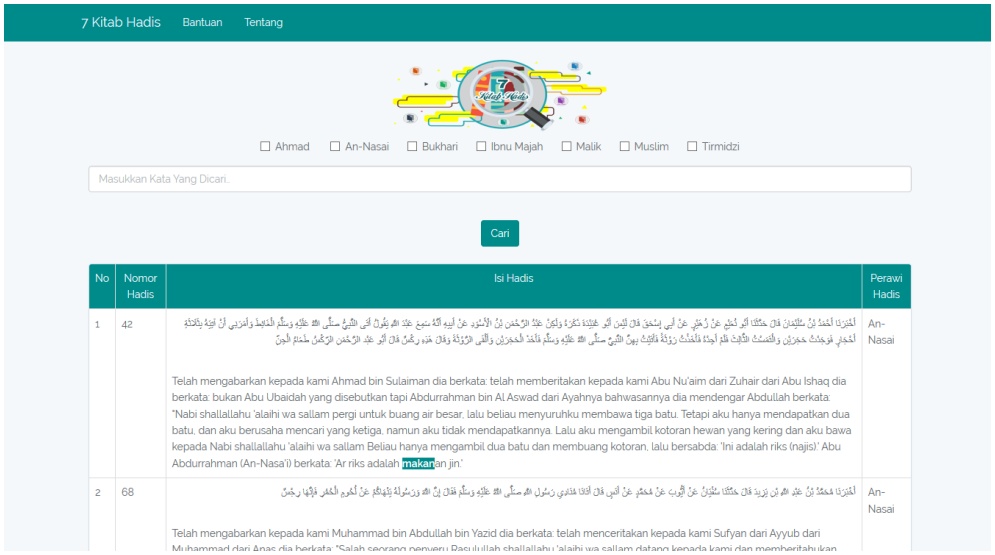
Pengguna melakukan pencarian dengan berbagai macam kata kunci yang akan dicari. Melihat dari data APJII, mesin pencari menjadi layanan ketiga terpopuler di bawah layanan *chatting* dan *social media*. Sebanyak 74,84 persen mengungguli layanan lihat gambar maupun lihat video. Itu artinya kehadiran layanan sistem pencari merupakan kebutuhan utama saat memulai berselancar di dunia maya, termasuk diantaranya oleh umat Muslim.

Umat Muslim dengan total populasi 1,5 miliar jiwa (Hackett *et al.*, 2015) saling berinteraksi tak hanya melalui percakapan langsung, namun juga sudah memanfaatkan teknologi yang ada. Dalam pelaksanaaannya, telah banyak menyisipkan teknologi ke sela-sela bidang, seperti dakwah yang dijalani oleh umat Muslim di dunia. Saat melakukan dakwah tentunya akan menyampaikan sumber hukum yang valid. Sumber hukum tersebut selain ada di Alquran, terdapat pula di Hadis yang merupakan sumber hukum kedua setelahnya. Sehingga dengan mengembangkan sistem pencarian tujuh Kitab Hadis tentunya akan sangat membantu umat Muslim dalam menemukan sumber hukum yang jelas dengan kemudahan akses yang diberikan.

Pengembangan sistem pencarian Hadis sebelumnya telah dikembangkan oleh Agustina (2018), penelitian tersebut membahas mengenai optimasi pencarian Hadis dalam empat kitab Hadis. Penelitian ini dilakukan untuk menyempurnakan hasil pencarian dari suatu kata kunci. Pencarian dilakukan dengan cara memberikan bobot pada masing-masing kata kunci yang akan

dicari. Dengan begitu, hasil pencarian Hadis dengan bobot besar akan tampil dalam halaman pertama pencarian.

Penelitian terakhir yang berkaitan dengan sistem pencarian Hadis dimiliki oleh Sari (2018). Penelitian tersebut membahas penggunaan Algoritma *string matching Knuth-Morris-Pratt* untuk melakukan pencarian pada tujuh kitab Hadis.



Gambar 1. Tampilan Halaman Hasil Pencarian Pada Sistem Pencarian Tujuh Kitab Hadis.

Dengan menggunakan algoritma tersebut, pencarian Hadis dengan kata kunci dapat dilakukan. Namun demikian, lama waktu pencarian sangat dipengaruhi oleh panjang teks pada *database*, semakin panjang teks maka semakin lama waktu yang dibutuhkan untuk mencari kata kunci. Selain itu, sistem ini belum mampu melakukan pencarian multi teks. Sistem hanya terpaku pada satu *pattern* kata kunci tanpa mencari kembali kata per kata. Saat ini mesin pencari telah dilengkapi dengan fitur *Spelling corrector*. Fitur itu dibutuhkan untuk mengoreksi kesalahan penulisan kata kunci. Namun pada penelitian

sebelumnya, sistem belum mampu menangani kesalahan pengetikan pada kata kunci yang diketik oleh pengguna. Melihat hal ini, penulis akan mengembangkan sistem pencarian Hadis multi teks pada tujuh kitab Hadis menggunakan Algortima *Knuth-Morris-Pratt.*

B. Rumusan Masalah

Rumusan masalah yang akan diselesaikan pada penelitian ini yaitu:

1. Bagaimana mengimplementasikan Algoritma *Knuth-Morris-Pratt* dalam pencarian multi teks pada tujuh kitab Hadis?
2. Bagaimana mendeteksi kesalahan penulisan kata kunci menggunakan *Norvig Spelling Corrector*?

C. Batasan Masalah

Batasan masalah penelitian ini adalah sebagai berikut:

1. Penelitian ini akan melakukan pencarian multi teks pada tujuh kitab Hadis yaitu Hadis Riwayat Ahmad, Bukhari, Ibnu Majah, Malik, Muslim, Nasa'i, dan Tirmidzi.
2. Penelitian ini menggunakan Algoritma *string matching Knuth-Morris-Pratt*.
3. Deteksi kesalahan penulisan kata kunci yang digunakan adalah *Norvig Spelling Corrector*.
4. Pencarian Hadis menggunakan Bahasa Indonesia.
5. Bahasa pemrograman yang digunakan adalah Python.
6. Sistem pencari Hadis dikembangkan menggunakan *framework* Django.

D. Tujuan

Adapun tujuan dari penelitian ini adalah:

1. Mengembangkan sistem pencarian muti kata pada tujuh kitab Hadis menggunakan Algoritma *string matching Knuth-Morris-Pratt*.
2. Mengembangkan sistem pencarian Hadis yang dapat mendeteksi kesalahan penulisan kata kunci menggunakan *Norvig Spelling Corrector*.

E. Manfaat

Manfaat dari penelitian ini adalah sebagai berikut:

1. Memberikan kemudahan dalam pencarian Hadis sebagai salah satu pedoman umat Muslim.
2. Memberikan informasi riwayat Hadis dari tujuh kitab Hadis.

II. TINJAUAN PUSTAKA

A. Sistem

Menurut Jogiyanto (2005), sistem merupakan sekumpulan elemen yang saling terkait untuk mencapai suatu tujuan tertentu. Komponen atau subsistem dari sebuah sistem memiliki dua atau lebih komponen yang berkaitan untuk berinteraksi satu sama lain.

Sistem memiliki dua kelompok pendekatan, yaitu sistem yang menekanan pada prosedur dan sistem yang menekankan pada elemennya. Pendekatan sistem yang menekankan pada prosedur merupakan suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkelompok, dan bekerja sama untuk melakukan kegiatan dalam menyelesaikan suatu sasaran tertentu.

Prosedur memiliki makna yakni urutan yang tepat dari tahapan-tahapan instruksi yang menerangkan apa *(what)* yang harus dikerjakan, siapa *(who)* yang mengerjakan, kapan *(when)* dikerjakan, dan bagaimana *(how)* mengerjakannya. Sedangkan pendekatan yang menekankan pada komponen atau elemen mendefinisikan sistem sebagai kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu.

Jogiyanto menjelaskan lebih lanjut, suatu sistem memiliki sifat-sifat atau karakteristik. Sifat-sifat tersebut diantaranya komponen-komponen *(components)*, batas sistem *(boundary)*, lingkungan luar sistem *(environments)*, penghubung *(interface)*, masukan *(input)*, keluaran *(output)*, pengolah *(process)*, dan sasaran *(objectives)* atau tujuan *(goal)*. Penjelasan mengenai komponen tersebut, yaitu:

1. Komponen Sistem

   Suatu sistem dalam mencapai tujuan tertentu memiliki komponen-komponen yang saling berkaitan dan bekerjasama membentuk satu kesatuan. Komponen yang dimaksud ialah subsistem atau bagian-bagian dari sistem. Setiap bagian-bagian sistem tersebut memiliki peran penting untuk menjalankan suatu fungsi atau tugas tertentu dan mempengaruhi proses sistem secara keseluruhan.

2. Batas Sistem

   Batas sistem merupakan daerah yang membatasi antara sistem dengan sistem lainnya, maupun dengan lingkungan luar sistem. Batas sistem bertujuan agar ruang lingkup dari sistem tersebut terlihat dengan jelas.

3. Lingkungan Luar Sistem

   Suatu sistem memiliki lingkungan luar sistem. Lingkungan luar sistem adalah apapun yang berada di luar batas dari sistem sehingga mempengaruhi operasi sistem.

4. Penghubung Sistem

   Media penghubung antara suatu subsistem dengan subsistem lainnnya disebut penghubung sistem. Media menghubungkan sumber daya

mengalir antar subsistem. Penghubung sistem berperan penting dalam media penghubung agar subsistem dengan subsistem lainnya dapat membentuk satu kesatuan.

5. Masukan Sistem

   Masukan *(input)* ialah energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan dan masukan sinyal. Masukan perawatan merupakan energi yang dimasukkan supaya sistem dapat berjalan. Sedangkan masukan sinyal adalah energi yang diproses untuk mendapatkan keluaran.

6. Keluaran Sistem

   Keluaran *(output)* merupakan hasil energi yang diolah sehingga menghasilkan keluaran yang berguna. Keluaran dapat berupa masukan untuk subsistem yang lain.

7. Pengolah Sistem

   Suatu sistem memiliki pengolah yang bertugas mengubah masukan menjadi keluaran. Sistem akuntansi akan mengolah data-data transaksi menjadi laporan-laporan keuangan dan laporan-laporan lain yang dibutuhkan manajemen.

8. Sasaran Sistem

   Suatu sistem pasti memiliki tujuan atau sasaran yang akan dicapai. Sasaran sistem sangat menentukan sekali masukan yang akan diolah oleh sistem dan dihasilkan keluaran oleh sistem. Jika suatu sistem tidak memiliki sasaran, maka operasi sistem tidak akan berguna.

B. *Search Engine*

Seymour *et al.* (2011) mengungkapkan *search engine* adalah program perangkat lunak yang mencari sekelompok situs *web* berdasarkan kata-kata yang ditentukan sebagai istilah pencarian (kata-kata pencarian). Mesin telusur melihatnya ke dalam *database* untuk menemukan informasi dari apa yang dicari.

Mesin pencari merupakan perangkat yang digunakan untuk mencari informasi dalam koleksi dokumen sistem. Pengguna hanya tinggal memasukkan kata-kata kunci dari informasi yang dicari, dan dalam kurun waktu tertentu sistem akan menampilkan hasil pencarian dari dokumen yang sesuai dengan kebutuhan informasi pengguna (Mandala, 2006).

Metode dari pencarian teks diantaranya:

1. *Keyword Searching*

   Sebagian mesin telusur melakukan pencarian menggunakan kata kunci. Namun metode ini juga kerap menemui masalah yang disebut *stemming*.

2. *Concept Searching*

   Sistem pencari berbasis konsep *(concept searching)* melakukan pencarian dengan mencoba memahami apa yang dimaksud, bukan hanya apa yang dikatakan. Konsep ini sudah banyak digunakan pada mesin telusur terpopuler saat ini.

3. *Meta Search Engine*

   Meta search engine adalah alat pencarian yang mengirimkan permintaan pengguna kebeberapa mesin pencari lain. Hasil yang didapat akan

ditampilkan sesuai dengan sumbernya. Metode ini tidak memiliki *database* halaman *web* (Seymour *et al.*, 2011).

Mesin pencari harus memiliki tiga jenis kebutuhan dibalik kata kunci yang dicari, diantaranya: informasi, navigasi, dan transaksi. Taknonomi tersebut sangat penting untuk pengembangan pencarian web yang sukses. Tujuan adalah untuk menangani secara efisien dari kueri dengan memahami apa yang dimaksud dari kata kunci tersebut (Broder, 2002).

C. Hadis

Hadis merupakan sabda dan perbuatan Nabi Muhammad Shallallahu Alaihi Wasallam yang diriwayatkan atau diceritakan oleh sahabat-sahabat Nabi untuk menjelaskan dan menentukan hukum Islam (Sugono, 2008).

Menurut Zen dan Khairiyah (2014), Hadis ialah perkataan, perbuatan, ketetapan, dan persetujuan dari Nabi Muhammad Shallallahu Alaihi Wasallam yang dijadikan landasan hukum Islam. Kedudukan Hadis berada sebagai sumber hukum kedua setelah Alquran.

D. *Spell Correction*

Zhu *et al*. (2012), mengungkapkan *Spell Correction* adalah suatu teknik dalam mengubah kesalahan ejaan kata atau frasa yang benar. Skenario *Spell Correction* dapat diterapkan dalam beragam kasus permasalahan, diantaranya: pengeditan dokumen, pencarian *suggestion*, koreksi kata kunci, dan lain-lain.

Spell Correction berfungsi sebagai asisten yang membantu pengguna untuk menghindari salah pengetikan.

Dalam proses *Spell Correction*, tak ada yang bisa mengetahui dengan pasti kebenaran ejaan kata. Namun dengan menggunakan probabilitas, kesalahan penulisan ejaan kata tersebut dapat dideteksi dari berbagai kandidat yang terpilih (Norvig, 2007).

E. *String Matching Algorithm*

Algoritma merupakan prosedur komputasi yang mentransformasi sejumlah *input* menjadi sejumlah *output*. Sebuah algoritma dikatakan benar jika untuk setiap masukan menghasilkan keluaran yang benar pula (Purbasari, 2007).

Purwanto (2008) dalam bukunya berpendapat bahwa Algoritma merupakan urutan operasi yang dilakukan terhadap data terorganisasi dalam struktur data. Selain itu pula, Algoritma disebut sebagai program abstrak yang dapat dieksekusi secara fisik oleh mesin.

Purwanto juga mengungkapkan bahwa *string matching* dalam Ilmu Komputer adalah suatu metode yang digunakan untuk mencocokkan lambang atau huruf *(string)* dalam suatu deret. Komputer akan mendapatkan satu pola atau deretan pola yang diberikan.

String matching algorithm adalah komponen dasar yang digunakan dalam implementasi perangkat lunak praktis yang ada pada sebagian besar sistem operasi (Charras dan Lecroq, 2004).

Purwanto (2008) menyatakan jika Algoritma *String Matching* merupakan metode yang dipakai untuk menemukan suatu keakuratan atau hasil dari satu atau beberapa pola teks yang diberikan.

F. Algoritma *Knuth-Morris-Pratt*

Algoritma *Knuth-Morris-Pratt* digunakan untuk menemukan semua kemungkinan dari panjang *pattern* dari sebuah teks, tanpa *input* kembali teks. Algoritma ini hanya membutuhkan lokasi *O(m)* dari memori internal jika teks dibaca dari berkas luar. Dasar dari terbentuknya algoritma ini dengan berimajinasi menempatkan *pattern* ke dalam teks dan menggeser *pattern* tersebut ke kanan dengan cara tertentu (Knuth *et al.*, 1977).

Algortima *Knuth Morris Pratt* untuk memproses dua karakter (atau *byte*) secara paralel, hal ini digunakan agar mempercepat proses pencocokan *pattern* (Ambika *et al.*, 2013).

Menurut Sarno *et al.* (2012), Algortima *Knuth-Morris-Pratt* merupakan proses pencocokan *string*. Bila terjadi ketidakcocokan pada saat *pattern* sejajar dengan teks *[i..i + n-1]*, dianggap bahwa ketidakcocokan pertama terjadi antara teks *[i+j]* dan *pattern [j]*, dengan $i<j<n$. Berarti, teks*[i..i+j]=pattern[0..j+1]* dan a=teks*[i+j]* tidak sama dengan b=*pattern[j]*, ketika digeser.

Pencocokan *string* akan berjalan efisien bila ada tabel yang menentukan berapa panjang untuk menggeser jika tidak terjadi kecocokan di karakter ke-*j* dari *pattern*. Tabel tersebut harus memuat *next[j]* yang merupakan posisi karakter *pattern[j]* setelah digeser, sehingga dapat menggeser *pattern* secara besar *j*-

next[j] relatif terhadap teks. Secara sistematis, langkah-langkah yang dilakukan algoritma *Knuth-Morris-Pratt* pada saat pencocokan *string* ialah sebagai berikut:

1. Algortima *Knuth-Morris-Pratt* mulai mencocokkan *pattern* dengan teks.
2. Dari kiri ke kanan, algortima akan mencocokkan karakter per karakter *pattern*, dengan karakter di teks yang bersesuaian sampai salah satu kondisi berikut terpenuhi:
   a. Karakter di *pattern* dan di teks yang dibandingkan tidak cocok *(mismatch)*
   b. Semua karakter di *pattern* cocok. Kemudian algortima akan memberitahukan penemuan di posisi ini.

Algorima kemudian menggeser *pattern* berdasarkan table *next*, lalu menghitung langkah 2 sampai *pattern* berada diujung teks (Mandala dan Waruwu, 2016).

Berikut merupakan proses dari algoritma *Knuth-Morris-Pratt* (Rahim *et al.*, 2017) untuk mencari sebuah kata dari teks, yaitu:

1. Diberikan sebuah variabel *S* bertipe *string*

2. Diberikan sebuah *pattern* dengan variabel *P* yang akan dicari dari teks di atas.

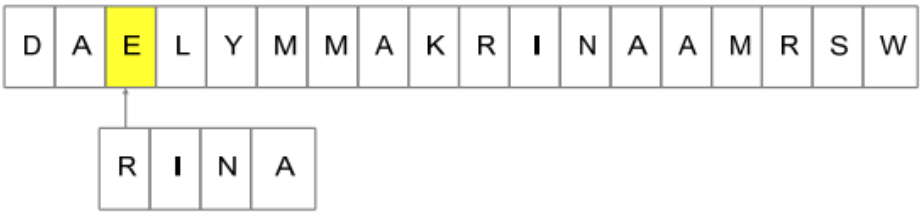
3. Langkah pertama, membandingkan *pattern P[1]* dengan *string S[1]*

Pattern[1] tidak sama dengan *S[1]*, lalu *pattern* akan bergeser satu posisi ke kanan.

4. Langkah kedua, membandingkan *pattern P[1]* dengan *string S[2]*

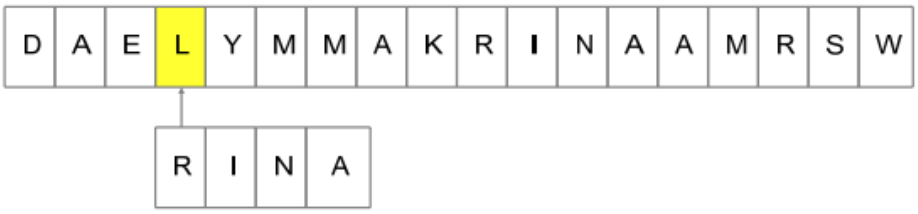
Pattern[1] tidak sama dengan *S[2]*, lalu *pattern* akan bergeser satu posisi ke kanan.

5. Langkah ketiga, membandingkan *pattern P[1]* dengan *string S[3]*



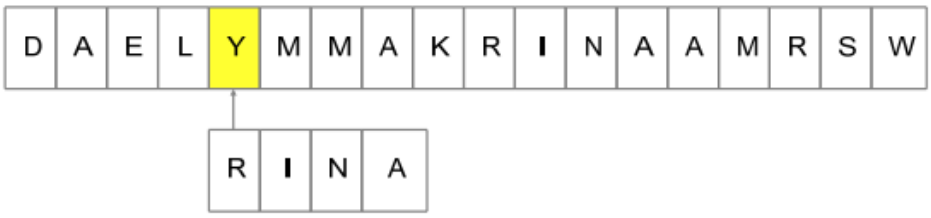
Pattern[1] tidak sama dengan *S[3]*, lalu pattern akan pindah satu posisi ke kanan.

6. Langkah keempat, membandingkan *pattern P[1]* dengan *string S[4]*



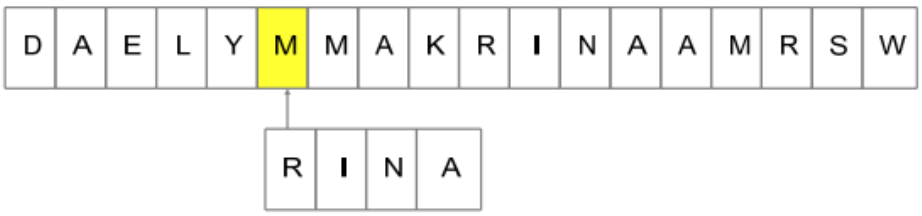
Pattern[1] tidak sama dengan *S[4]*, lalu *pattern* akan pindah satu posisi ke kanan.

7. Langkah kelima, membandingkan *pattern P[1]* dengan *string S[5]*



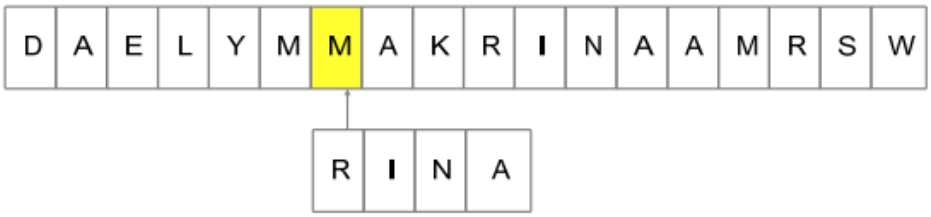
Pattern[1] tidak sama dengan *S[5]*, lalu *pattern* akan bergeser satu posisi ke kanan.

8. Langkah keenam, membandingkan *pattern P[1]* dengan *string S[6]*



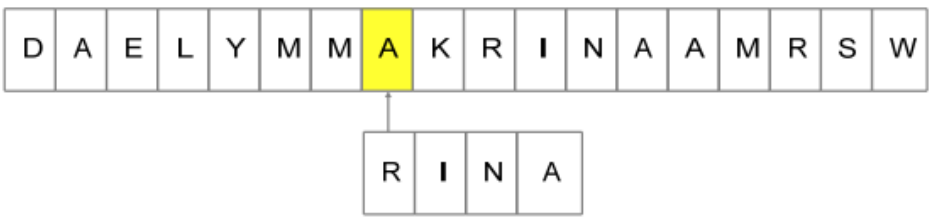
Pattern[1] tidak sama dengan *S[6]*, lalu *pattern* akan bergeser satu posisi ke kanan.

9. Langkah ketujuh, membandingkan *pattern P[1]* dengan *string S[7]*



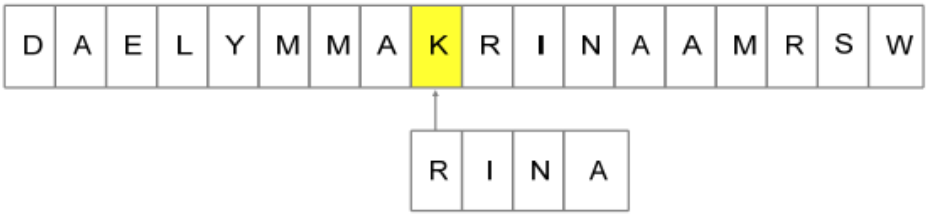
Pattern[1] tidak sama dengan *S[7]*, lalu *pattern* akan pindah satu posisi ke kanan.

10. Langkah kedelapan, membandingkan *pattern P[1]* dengan *string S[8]*



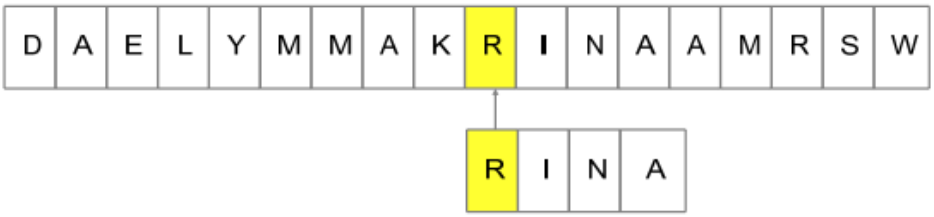
Pattern[1] tidak sama dengan *S[8],* lalu *pattern* akan pindah satu posisi ke kanan.

11. Langkah kesembilan, membandingkan *pattern P[1]* dengan *string S[9]*

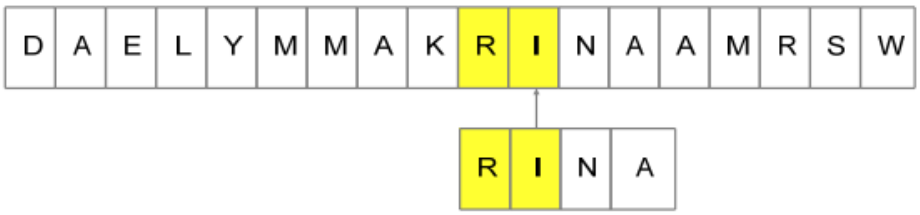


Pattern[1] tidak sama dengan *S[9],* lalu *pattern* akan pindah satu posisi ke kanan.

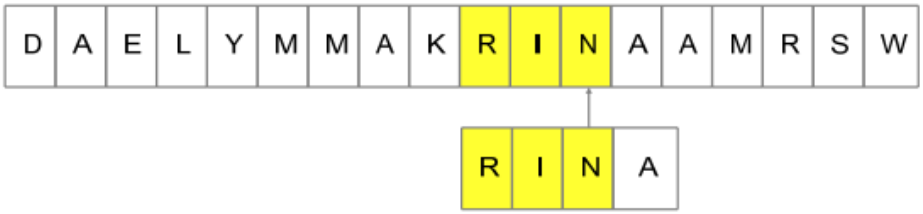
12. Langkah kesepuluh, membandingkan *pattern P[1]* dengan *string S[10]*



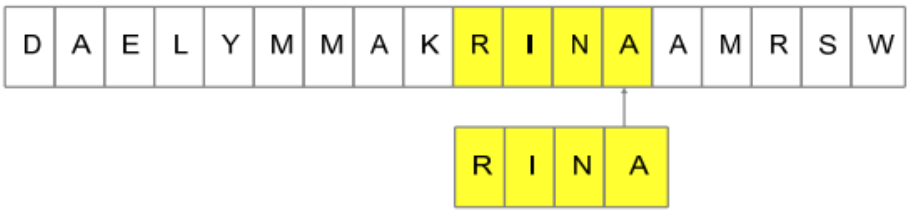
Pattern[1] sama dengan *S[10]*, karena ada kecocokan, algoritma *Knuth-Morris-Pratt* akan menyimpan informasi ini, dan *pattern* tidak bergeser namun melanjutkan membandingkan *pattern[2]* dengan *S[11]*.



Pattern[1,2] cocok dengan *S[10,11]*. Karena sama, algoritma *Knuth-Morris-Pratt* akan menyimpan informasi ini, dan *pattern* tidak bergeser namun melanjutkan membandingkan *pattern[3]* dengan *S[12]*.



Pattern[1,2,3] cocok dengan *S[10,11,12]*. Karena cocok, Algoritma *Knuth-Morris-Pratt* akan menyimpan informasi ini, dan *pattern* tidak bergeser namun melanjutkan membandingkan *pattern[4]* dengan *S[14]*.



Pattern[1,2,3,4] cocok dengan *S[10,11,12,13]*. Karena cocok, algoritma *Knuth-Morris-Pratt* akan menyimpan informasi ini, dan *pattern* tidak bergeser namun melanjutkan membandingkan *pattern[5]* dengan *S[15]*. Namun karena jumlah *pattern* hanya empat huruf, maka pencarian dihentikan dan hasilnya adalah *pattern P* cocok dengan *sting S* sebesar 100 persen.

G. Python

Python adalah bahasa pemrograman modern yang mudah dipelajari dan mudah digunakan. Python juga merupakan bahasa pemrograman yang ideal, karena menggabungkan sintaks yang jelas dan sederhana dengan dukungan untuk satu set struktur kontrol dan tipe data bawaan yang kuat. Selain itu, Python digunakan oleh banyak *programmer* profesional untuk menulis sistem perangkat lunak yang kompleks (Miller, 2008).

H. *Hyper Text Markup Language*

Dokumen *Hyper Text Markup Language* (HTML) adalah *file* teks murni yang dapat dibuat dengan editor teks sembarang. Dokumen ini dikenal sebagai *web page*. *File* HTML ini berisi instruksi-instruksi yang kemudian diterjemahkan oleh *browser* yang ada di komputer *client (user)* sehingga isi informasinya dapat ditampilkan secara visual di komputer pengguna (Kustiyahningsih dan Anamisa, 2011).

HTML dikenal sebagai standar bahasa yang digunakan untuk menampilkan dokumen web. Hal-hal yang bisa dilakukan HTML yaitu:

1. Mengontrol tampilan dari *web page* dan kontennya.
2. Mempublikasikan dokumen secara *online* sehingga bisa diakses dari seluruh dunia.
3. Membuat *online form* yang bisa digunakan untuk menangani pendaftaran, transaksi secara *online*.
4. Menambahkan objek, seperti *image*, *audio, video* dan juga *Java applet*.

I. Django

Django adalah *framework* sumber terbuka dan gratis yang ditulis dengan Python. Django memiliki pola arsitektur *Model-View-Template* (MVT). Tujuan utama Django adalah untuk mempermudah pembuatan situs yang kompleks dan didorong oleh *database*. Django menekankan *reusability* dan *pluggability* dari komponen, sedikit kode, dan mudah dikembangkan dengan

cepat. Django juga menyediakan opsional untuk membuat halaman administratif, seperti membaca, memperbarui dan menghapus antarmuka yang dihasilkan secara dinamis dan dikonfigurasi melalui model *admin* (Holovaty dan Moss, 2006).

J. *Database*

Menurut Connolly dan Begg (2005), *database* adalah kumpulan data yang berhubungan secara logis dan dirancang untuk memenuhi kebutuhan informasi dari sebuah organisasi.

Sedangkan menurut Whitten *et al.* (2004), *database* adalah kumpulan *file* yang saling terkait. Tidak hanya kumpulan *file*, tetapi *record* pada setiap *file* harus saling berhubungan untuk menyimpan *file* lain.

Basis data *(database)* merupakan suatu pengorganisasian sekumpulan data yang saling terkait sehingga memudahkan aktivitas untuk memperoleh informasi (Kadir, 2014).

Basis data *(database)* dibuat untuk mengatasi permasalahan yang terdapat pada suatu sistem dengan menggunakan pendekatan berbasis kelas. Berbeda dengan sistem berkas atau sistem *file* yang menyimpan data secara terpisah, sebuah data pada basis data *(database)* tersimpan secara terintegrasi (Ariani dan Shalahudin, 2011).

Ariani dan Shalahudin menambahkan, untuk mengelola basis data *(database)* diperlukan sebuah perangkat lunak yang disebut *Database Management*

System (DBMS). DBMS merupakan sebuah sistem yang digunakan untuk menyimpan, mengelola, dan menampilkan data. Berikut merupakan komponen penyusun utama dari sebuah basis data *(database)*:

1. *Hardware* (Perangkat Keras)

   Komponen ini berupa perangkat komputer standar, media penyimpan sekunder dan media komunikasi untuk sistem jaringan.

2. *Operating System* (Sistem Operasi)

   Komponen ini merupakan perangkat lunak yang berfungsi untuk mengendalikan seluruh sumber daya dan melakukan operasi dasar dalam sistem komputer.

3. *Database* (Basis Data)

   Komponen ini merupakan basis data yang mewakili sistem tertentu untuk dikelola. Sebuah sistem basis data bisa terdiri dari lebih dari satu basis data.

4. *Database Management System*

   Komponen ini merupakan perangkat lunak yang digunakan untuk mengelola basis data.

5. *User* (Pengguna sistem basis data)

   Komponen ini merupakan orang yang berinteraksi dengan sistem basis data, mulai dari perancang sampai dengan pengguna tingkat akhir.

6. *Optional Software* (Perangkat Lunak Opsional)

   Komponen ini merupakan perangkat lunak pelengkap yang mendukung. Bersifat abstraksi data pada sebuah basis data yang merupakan level penjelasan cara untuk melihat data dalam sebuah sistem basis data. Pada umumnya pengguna hanya mengerti bagaimana cara sebuah data dapat

terlihat tetapi tidak mengetahui bagaimana data tersebut disimpan dan dipelihara. Abstraksi data pada basis data terdiri dari tiga level yaitu level eksternal, level konseptual, dan level internal.

K. Apache

Menurut Delisle (2008), Apache adalah server *web* yang dapat dijalankan di banyak sistem operasi (*Unix, BSD, Linux, Microsoft Windows,* dan *Novell Netware,* serta platform lainnya) yang berguna untuk melayani dan memfungsikan situs *web*. Protokol yang digunakan untuk melayani fasilitas *web/www* ini mengunakan HTTP. Apache memiliki fitur-fitur canggih seperti pesan kesalahan yang dapat dikonfigurasi, autentikasi berbasis basis data dan lain-lain. Apache juga didukung oleh sejumlah antarmuka pengguna berbasis grafik (GUI) yang memungkinkan penanganan *server* menjadi mudah.

Apache merupakan perangkat lunak sumber terbuka dikembangkan oleh komunitas terbuka yang terdiri dari pengembang-pengembang di bawah naungan *Apache Software Foundation.*

L. MariaDB *Server*

Server MariaDB adalah salah satu *server* basis data yang ada di dunia. MariaDB Dibuat oleh pengembang asli *My Structured Query Language* (MySQL) dan *open source*. MariaDB mengubah data menjadi informasi terstruktur dalam beragam aplikasi, mulai dari perbankan hingga situs *web*. MariaDB dikembangkan sebagai perangkat lunak *open source* dan sebagai

database relational yang menyediakan antarmuka SQL untuk mengakses data (Dyer, 2015). Berdasarkan survei Stackoverflow (2018), MariaDB masuk dalam jajaran sepuluh besar *database* yang paling banyak digunakan oleh *programmer* dunia.

M. Pengujian Sistem

Pengujian sistem merupakan proses untuk mengecek apakah suatu perangkat lunak yang dihasilkan sudah dapat dijalankan sesuai standar atau belum. Pengecekan program aplikasi dilakukan dengan pengecekan *input*, pengecekan proses dan pengecekan *output* (Yakub, 2012), yaitu:

a. Pengecekan *input*, meliputi kelengkapan item-item *input*, kemudahan pengoperasian, kemudahan manipulasi data, dan pengendalian kesalahan.
b. Pengecekan proses, dilakukan sekaligus dengan pengecekan *output* program.
c. Pengecekan *output*, meliputi pengecekan terhadap format dan bentuk-bentuk laporan.

N. *Black Box Testing*

Pengujian *black box* merupakan suatu teknik pengujian perangkat lunak dengan berfokus pada persyaratan fungsional. Pengujian *black box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi *input* yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Pengujian *black box* berusaha menemukan kesalahan dalam kategori yaitu:

1. Fungsi-fungsi yang tidak benar atau hilang.
2. Kesalahan *interface*.
3. Kesalahan dalam struktur data atau akses *database* eksternal.
4. Kesalahan kinerja.
5. Inisialisasi dan kesalahan terminasi.

Pengujian *black box* diaplikasikan selama tahap akhir pengujian karena *black box* memperhatikan stuktur kontrol, maka perhatian berfokus pada domain informasi (Pressman, 2010).

Equivalence Partitioning merupakan metode *black box testing* yang membagi domain masukan dari program ke dalam kelas-kelas sehingga *test case* dapat diperoleh. *Equivalence Partitioning* berusaha untuk mendefinisikan kasus uji yang menemukan sejumlah jenis kesalahan, dan mengurangi jumlah kasus uji yang harus dibuat. Kasus uji yang didesain untuk *Equivalence Partitioning* berdasarkan pada evaluasi dari kelas ekuivalensi untuk kondisi masukan yang menggambarkan kumpulan keadaan yang benar atau tidak. Kondisi masukan dapat berupa spesifikasi nilai numerik, kisaran nilai, kumpulan nilai yang berhubungan.

Ekuivalensi *class* dapat didefinisikan dengan panduan berikut:

1. Jika kondisi *input* menspesifikasikan kisaran/*range*, maka didefinisikan 1 yang valid dan 2 yang invalid untuk *equivalence class*,
2. Jika kondisi *input* memerlukan nilai yang spesifik, maka didefinisikan 1 yang valid dan 2 yang invalid untuk *equivalence class*,

3. Jika kondisi *input* menspesifikasikan anggota dan himpunan, maka didefinisikan 1 yang valid dan 2 yang invalid untuk *equivalence class*,
4. Jika kondisi *input* adalah *boolean*, maka didefinisikan 1 yang valid dan 2 yang invalid untuk *equivalence* class (Pressman, 2010).

O. Skala *Likert*

Skala *Likert* adalah metode penskalaan pertanyaan sikap yang menggunakan distribusi respon sebagai dasar penentuan nilai skalanya. Nilai skala setiap pernyataan ditentukan oleh distribusi respon setuju dan tidak setuju dari sekelompok responden yang bertindak sebagai kelompok uji coba (Azwar, 2011).

Skala *Likert* digunakan untuk mengukur pendapat dan sikap dari setiap responden. Pada skala pengukuran ini dinyatakan dalam beberapa pilihan jawaban seperti baik, tidak baik, sangat baik, dan lain-lain atau dapat berupa nilai rentang antara 0-5, 1-5, 1-3, dan sebagainya dengan keterangan nilai mana yang memiliki *value* tertinggi dan nilai mana yang memiliki *value* terendah.

Kelebihannya adalah responden dapat memberikan pendapat dan penilainnya dengan pilihan yang sesuai, sehingga jawaban yang diberikan pun bervariasi, mudah dibuat dan diterapkan. Kekurangannya adalah dengan banyak pilihan yang diberikan kepada responden akan bingung terhadap jawaban atau pendapat yang akan diberikan. Dengan banyaknya pilihan juga membutuhkan waktu yang cukup lama untuk menjawab. Jawaban tidak tegas karena semua

jawaban didasarkan pada pendapat masing-masing responden (Islamiah, 2014).

Azwar menambahkan, Skala *Likert* berisi lima tingkat prefensi jawaban dengan pilihan sebagai berikut:

1. Tidak setuju
2. Kurang Setuju
3. Cukup Setuju
4. Setuju
5. Sangat setuju

Penentuan kategori tersebut menggunakan rumus sebagai berikut:

$$P = \frac{Xi}{n\ x\ N}\ x\ 100\%$$

Keterangan:

P = Persentase pernyataan,

Xi = Nilai kuantitatif total,

n = Jumlah responden,

N = Nilai kategori tertinggi.

Selanjutnya, penentuan interval per kategori digunakan rumus sebagai berikut:

$$I = \frac{100\%}{K}$$

Keterangan:

I = Interval,

K = Kategori Interval.

P. Penelitian Terkait

Hasil penelitian yang relevan dengan penelitian ini adalah:

1. Penelitian dilakukan oleh Sidi (2017) mengenai pengembangan sistem pencarian informasi pada Hadis Riwayat Bukhari. Penelitian ini bertujuan untuk membantu pengguna dalam mencari Hadis riwayat Bukhari, serta membangun sistem pencarian alternatif dari mesin pencari yang sudah ada. Total data yang digunakan sebanyak 7008 data Hadis. Penelitian ini menyimpulkan bahwa sistem pencarian dapat menampilkan nomor dan isi Hadis sehingga memudahkan pengguna dalam mendapatkan informasi Hadis. Selain itu, sistem *database* yang dibangun belum menggunakan *index*.
2. Penelitian berjudul "*Perbandingan Pencarian Hadis Dengan dan Tanpa Index pada Sistem Pencarian Hadis Riwayat Imam Malik dan Bukhari*" oleh Ziyadurrohman dilakukan pada tahun 2017. Penelitian ini bertujuan untuk membandingkan pengaruh penggunaan *index* dan tanpa *index* dalam proses pencarian. Hadis riwayat Imam Malik dan Bukhari digunakan untuk membandingkan pengaruh tersebut. Penelitian ini menarik kesimpulan bahwa penggunaan *index* menunjukkan waktu yang dibutuhkan lebih cepat jika dibandingkan tanpa menggunakan *index*. Hal ini terlihat dari lama waktu sistem dalam melakukan pencarian ke dalam *database*.
3. Penelitian sebelumnya juga dilakukan oleh Agustina tahun 2018. Optimasi pencarian Hadis dalam empat kitab Hadis dilakukan bertujuan untuk mengoptimalkan proses pencarian pada empat kitab Hadis sehingga membantu pengguna menemukan informasi. Dalam penelitian ini,

Agustina melakukan pembobotan tiap kata yang dicari agar hasil pencarian terletak pada halaman pertama. Hasil yang didapat, sistem yang dikembangkan sudah cukup optimal dalam melakukan pencarian Hadis.

4. Penelitian oleh Sari pada 2018 lalu juga membahas mengenai sistem pencari Hadis. Topik yang diteliti mengenai pengembangan sistem pencarian dari tujuh kitab Hadis menggunakan algortima *Knuth-Morris-Pratt*. Tujuannya untuk mengimplementasikan algorima pencarian *string Knuth-Morris-Pratt* dalam melakukan pencarian dari tujuh kitab Hadis. Dari hasil penelitian ini disimpulkan bahwa sistem ini dapat menampilkan isi Hadis berdasarkan orang yang meriwayatkan Hadis (perawi). Data yang digunakan pada penelitian ini menggunakan kitab Ahmad dengan jumlah data 26.363 Hadis, kitab An-nasa'i dengan jumlah data 5.662 Hadis, kitab Bukhari dengan jumlah data 7.008 Hadis, kitab Ibnu Majah dengan jumlah data 4.332 Hadis, kitab Malik dengan jumlah data 1.594 Hadis, kitab Muslim dengan jumlah data 5.362 Hadis, dan kitab Tirmidzi dengan jumlah data 3.891 Hadis. Selain itu, disimpulkan pula bahwa panjang teks dari *database* dan *pattern* yang dicari akan mempengaruhi lama waktu pencarian.

III. METODOLOGI PENELITIAN

A. Tempat dan Waktu Penelitian

Penelitian ini dilakukan di Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung. Waktu penelitian dilakukan pada Semester Ganjil Tahun Ajaran 2018/2019 dengan sumber Hadis didapat dari Al-Islam dan Hadis Bot pada aplikasi Telegram.

B. Bahan dan Alat Penelitian

Bahan yang digunakan dalam penelitian ini adalah data Hadis. Berikut penulis jabarkan data Hadis yang digunakan dalam penelitian ini, yaitu:

1. Kitab Ahmad dengan jumlah 26.363 Hadis.
2. Kitab Bukhari dengan jumlah 7.008 Hadis.
3. Kitab Ibnu Majah dengan jumlah 4.332 Hadis.
4. Kitab Malik dengan jumlah 1.594 Hadis.
5. Kitab Muslim dengan jumlah 5.362 Hadis.
6. Kitab Nasa'i dengan jumlah 5.662 Hadis.
7. Kitab Tirmidzi dengan jumlah 3.891 Hadis.

Total keseluruhan pada tujuh kitab Hadis adalah 52.618 Hadis. Hadis tersebut didokumentasikan ke dalam sebuah *database*. *Database* Hadis tersebut berasal

dari Sari (2018) yang telah mendokumentasikan tujuh kitab Hadis ke dalam *database*. Masing-masing perawi dimasukkan dalam sebuah tabel. Selain itu, peneliti juga menggunakan *data training* yang dari beberapa sampel isi Hadis Indonesia berdasarkan masing-masing perawi. *Data training* tersebut didokumentasikan ke dalam *file* txt. Jumlah *data training* yang peneliti gunakan sebesar 15.826 kata.

Penelitian ini dilakukan dengan menggunakan *hardware* dan *software* yang berguna untuk mendukung dan menunjang pelaksanaan penelitian, yaitu:

1. Perangkat keras (*Hardware*)

   Perangkat keras yang digunakan dalam pengembangan sistem ini adalah 1 unit *Notebook* dengan spesifikasi:

   a. *Processor* AMD *Quad-Core* A10-8700P 2,3 GHz
   b. *Display*: 15.6 inci, IPS FHD (1080x640)
   c. *Memori* 4GB RAM
   d. VGA: ATI *Radeon* R6
   e. *Storage*: 1 TB HDD

2. Perangkat lunak (*Software*)

   Perangkat lunak yang digunakan dalam pengembangan sistem ini adalah :

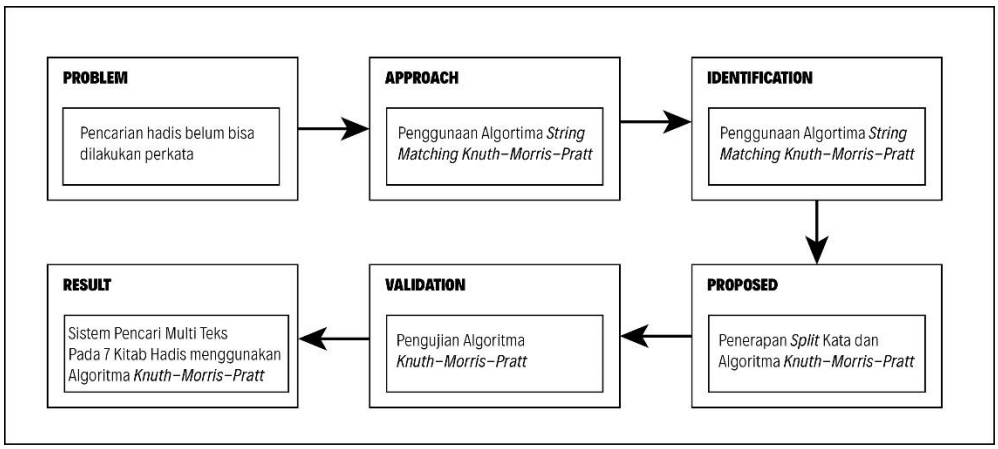
   a. Sistem operasi: Windows 10 Education
   b. *Web browser*: Mozilla Firefox
   c. Program editor: Visual Studio Code
   d. *Database server*: MariaDB
   e. *Web server*: Apache 2.4

f. Bahasa pemrograman: Python 3.0

g. *Framework* Django 12

C. Kerangka Penelitian

Kerangka penelitian ini akan mengambarkan permasalahan dari awal yang akan diteliti hingga mencapai hasil yang akan diperoleh. Kerangka penelitian dapat dilihat pada Gambar 2.



Gambar 2. Kerangka Penelitian.

Berikut ini penjelasan dari masing-masing tahap pada Gambar 3, yaitu:

1. Permasalahan Penelitian

   Permasalahan pada penelitian ini yaitu menyelesaikan pencarian multi teks menggunakan Algoritma *Knuth-Morris-Pratt*. Pada sistem sebelumnya, pencarian kata kunci hanya berpacu pada kata kunci tersebut, artinya kata kunci yang diketik oleh pengguna dianggap satu buah *pattern*. Sedangkan pada nyatanya, kata kunci yang diketik oleh pengguna besar kemungkinan terdiri dari banyak kata. Misalnya pada pencarian “makan daging”, pada sistem sebelumnya kata kunci tersebut dianggap satu buah *pattern* yang

akan dicocokkan ke dalam *database* Hadis. Namun semestinya kata kunci tersebut terdiri dari dua buah kata. Contoh lain jika mencari dengan kata kunci “makan daging sambil berdiri menggunakan tangan kiri”, sangat besar kemungkinan data tidak ditemukan di dalam *database* Hadis, karena sistem menganggap kata kunci tersebut sebagai satu *pattern*. Oleh sebab itu peneliti ingin mengembangkan sistem pencarian tersebut agar bisa memaksimalkan hasil pencarian menggunakan algortima *string matching* yang sama.

2. Analisis Pendekatan

   Analisis pendekatan dilakukan untuk menemukan solusi dari permasalahan yang dihadapi. Pendekatan yang dilakukan dengan menggunakan metode algoritma *Knuth-Morris-Pratt*. Salah satu algoritma *string matching* yang sering dipakai untuk menemukan suatu *pattern* dalam teks. Algortima ini bekerja melakukan pencarian *pattern* dari kiri ke kanan.

3. Identifikasi Masalah

   Identifikasi masalah merupakan tahapan yang akan dikerjakan dalam pengembangan sistem pencarian menggunakna algortima *Knuth-Morris-Pratt* pada tujuh kitab Hadis. Tahap permasalahan penelitian dan analisis pendekatan melandasi pertimbangan untuk mengidentifikasi masalah. Selanjutnya saat masalah sudah teridentifikasi dilakukan pemecahan masalah.

4. Pengajuan Solusi

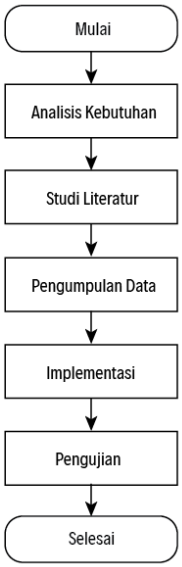
   Tahap pengajuan solusi berlandaskan identifikasi masalah yang telah dijabarkan. Solusi-solusi akan diberikan untuk menyelesaikan masalah yang ada.

5. Validasi

   Validasi merupakan tahap terakhir dari kerangka penelitian. Algoritma *string matching Knuth-Morris-Pratt* diimplementasikan ke dalam sistem pencari. Selain itu dilakukan juga tahap pemecahan kata kunci menjadi beberapa kata inti.

D. Tahapan Penelitian

Penelitian ini dilakukan beberapa tahap yaitu analisis kebutuhan, studi literatur, pengumpulan data, implementasi, dan pengujian. Tahapan penelitian dapat dilihat dari Gambar 3.



Gambar 3. Tahap Penelitian.

Berikut ini penjelasan dari masing-masing tahapan, yaitu:

1. Analisis Kebutuhan

   Analisis kebutuhan merupakan cara untuk mengidentifikasi, mengumpulkan studi literatur mengenai metode-metode *string matching* khusus Algortima *Knuth-Morris-Pratt*.

2. Studi Literatur

   Penelitian ini mengumpulkan data dan informasi yang dibutuhkan oleh sistem, melihat kekurangan sistem yang sudah ada, dan menyesuaikan sistem dengan kebutuhan pengguna berdasarkan data yang telah diperoleh.

3. Pengumpulan Data

   Penelitian ini mengambil data Hadis dari Al-Islam dan Hadis Bot yang telah disusun di dalam *database* oleh Desy Kartika Sari. Data korpus yang digunakan mengambil dari sampel Hadis Indonesia berdasarkan perawi masing-masing.

4. Implementasi

   Implementasi ialah tahap dalam menenerapkan algoritma *Knuth-Morris-Pratt* kedalam bahasa pemrograman Python. Penerapan algorima ini berdasarkan cara kerja yang terjadi pada algortima *Knuth-Morris-Pratt*.

5. Pengujian

   Tahap pengujian merupakan tahap akhir dari tahap-tahap yang sudah dilalui sebelumnya. Proses pencarian menggunakan algoritma *Knuth-Morris-Pratt* diukur dari hasil yang ditampilkan dari proses tersebut. Proses

melakukan *split* kata kunci, lalu melakukan pencarian teks ke dalam *database*, dan kembali menampilkannya dihasil pencarian.

E. Implementasi Algoritma *Knuth-Morris-Pratt*

Algoritma *Knuth-Morris-Pratt* memiliki dua tahapan dalam melakukan pencarian. Tahap pertama adalah *Pre-Processing the Pattern*. Tahap kedua yaitu *Searching the Pattern*. *Pre-Processing the Pattern* merupakan tahapan yang dibutuhkan sebelum melakukan pencarian utama *string*. Tahap ini mencari nilai dari pergeseran suatu *pattern*. Nilai pergeseran *pattern* ini dibutuhkan dalam melakukan pencarian *string* pada tahap kedua. Tahap inilah yang menentukan berapa banyak perpindahan *pattern* dari kiri ke kanan. Sedangkan tahap *Searching the Pattern* merupakan tahapan utama dalam melakukan pencarian *pattern* dari suatu teks. Tidak seperti *Naive Algorithm*, dimana proses pergeseran pola dengan cara satu per satu, *Knuth-Morris-Pratt Algorithm* menggunakan nilai dari *longest prefix suffix* atau *lps[]* untuk memutuskan besar pergeseran. Proses algoritma dijelaskan seperti berikut ini:

1. *Pre-Processing Algorithm*

   Pada bagian *preprocessing*, nilai *lps[]* dihitung dengan melacak panjang dari nilai awalan dan akhiran suatu *pattern*. Nilai tersebut didapat menggunakan variabel *len*. Nilai 0 diinisialisasi untuk *lps[0]* dan *len*. Kode program dari *Pseudocode Pro-Processing* ditunjukkan dalam Kode Program 1.

```
      procedure computeLPSArray(
      input pattern : array of string pattern,
      input panjangPattern : integer,
      input lps : array[0..m] of integer)

      Deklarasi:
      len : 0
      lps[0]
      i : 1

      Deskripsi:
      While (i < panjangPattern):
            If (pattern[i] == pattern[len])
                  len := len + 1
                  lps[i] := len
                  i := i + 1
            Else
                  If (Len != 0)
                        len = lps[len-1]
                  Else
                        lps[i] = 0
                        i := i +1
                  Endif
            Endif
    Endwhile
```

Kode Program 1. *Pseudocode Pre-Processing.*

Ilustrasi pencarian nilai *lps[]* dari *pattern* “AAAA” dapat dilihat melalui beberapa tahapan, yaitu:

Tahap pertama, *len* = 0 dan *i* = 0. *Lps[0]* selalu bernilai 0. Lalu pindah *i* = 1.

Tahap kedua, *len* = 0 dan *i* = 1. Karena *pat[len]* sama dengan *pat[i]*, maka nilai *len* dinaikkan, sehingga menjadi *len*=1. Selanjutnya nilai *lps[i]* disimpan, lalu nilai *i* dinaikan menjadi *i* = 2. Didapatkan nilai *lps[1]* = 1.

Tahap ketiga, *len* = 1 dan *i* = 2. Karena *pat[len]* sama dengan *pat[i]*, maka nilai *len* dinaikkan, sehingga menjadi *len*=2. Selanjutnya nilai *lps[i]* disimpan, lalu nilai *i* dinaikan menjadi *i* = 3. Didapatkan nilai *lps[1]* = 2.

Tahap keempat, *len* = 2 dan *i* = 3. Karena *pat[len]* sama dengan *pat[i]*, maka nilai *len* dinaikkan, sehingga menjadi *len*=3. Selanjutnya nilai *lps[i]* disimpan, lalu nilai *i* dinaikan menjadi *i* = 4. Didapatkan nilai *lps[1]* = 3. Berdasarkan pencarian nilai *lps[]* di atas, didapatkan bahwa *pattern* "AAAA" memiliki nilai *lps[]* = [0,1,2,3].

2. *Searching The Pattern*

Tahap *Searching the Pattern* merupakan tahap utama dalam pencarian *string*. Dalam tahap ini, nilai *lps[]* berperan untuk memutuskan posisi pergeseran selanjutnya. Kode program *Searching the Pattern* dapat dilihat melalui Kode Program 2.

```
procedure KMPSearch(input pattern : array,
input text :array)
Deklarasi:
M , N, j, i : integer
lps , hasil : array of result
Deskripsi:
M := length of pattern
N := length of text
lps := [0]*M
j := 0
computeLPSArray(pattern, M, lps)
i := 0
While (i < N):
      If (pattern[i] == pattern[len])
            i := i + 1
            j := j + 1
      IF (j == M)
            insert to hasil i-j
            j := lps[j-1]
      Else If (i < N and pat[j] != txt[i])
            if j != 0:
                j := lps[j-1]
            else:
                i := i + 1
            Endif
      Endif
Endwhile
```

Kode Program 2. *Pseudocode Searching the Pattern*

Proses pencarian *pattern* dimulai dengan membandingkan *pat[j]* dengan nilai *j* = 0. Selanjutnya karakter *txt[i]* dan *pat[j]* terus dilakukan pencocokan dengan terus menambah nilai *i* dan nilai *j*. Saat terjadi ketidakcocokan, pasti diketahui nilai *pat[0..j-1]* dan nilai *txt[0..i-1]* dan nilai *j* akan mengalami kenaikan jika terjadi kecocokan. Berikut ilustrasi proses pencarian *pattern* di suatu teks yang ada.

Tabel 1. Pencocokan *Pattern* ke dalam *Text*

Index	0	1	2	3	4	5	6	7	8	9	10
Text	**A**	**A**	**A**	**A**	**A**	**B**	**A**	**A**	**A**	**B**	**A**

Index	0	1	2	3
Pattern	**A**	**A**	**A**	**A**

Tabel 1 menunjukkan inisialisasi *Text* dan *Pattern*. *Text* mengilustrasikan teks yang berada di dalam *database*. Sedangkan *Pattern* mengilustrasikan kata kunci yang ingin dicari atau dicocokkan ke dalam teks. Berdasarkan tahap *Pre-Processing* sebelumnya, nilai *lps[]* adalah *[0, 1, 2, 3]*.

Tabel 2. Mencocokkan *Pattern* ke *Text* pada indeks 0

Index	0	1	2	3	4	5	6	7	8	9	10
Text	**A**	**A**	**A**	**A**	**A**	**B**	**A**	**A**	**A**	**B**	**A**

Index	0	1	2	3
Text	**A**	**A**	**A**	**A**

Tabel 2 menunjukkan perbandingan karakter *Pattern* dan karakter pada *Text*. *Pattern* indeks 0 atau *pat[0]* dicocokkan dengan *Text* indeks ke 0 atau *txt[0]*. Setelah dicocokkan, diketahui bahwa kedua karakter cocok. Lalu lanjut ke indeks *i* = 1 dan *j* = 1.

Tabel 3. Mencocokkan *Pattern* ke *Text* pada indeks 1

Index	0	1	2	3	4	5	6	7	8	9	10
Text	**A**	**A**	**A**	**A**	**A**	**B**	**A**	**A**	**A**	**B**	**A**

Index	0	1	2	3
Text	**A**	**A**	**A**	**A**

Tabel 3 menunjukkan perbandingan karakter *Pattern* dan karakter pada *Text*. *Pattern* indeks 1 atau *pat[1]* dicocokkan dengan *Text* indeks ke 1 atau *txt[1]*. Setelah dicocokkan, diketahui bahwa kedua karakter cocok. Lalu lanjut ke indeks $i = 2$ dan $j = 2$.

Tabel 4. Mencocokkan *Pattern* ke *Text* pada indeks 2

Index	0	1	2	3	4	5	6	7	8	9	10
Text	**A**	**A**	**A**	**A**	**A**	**B**	**A**	**A**	**A**	**B**	**A**

Index	0	1	2	3
Text	**A**	**A**	**A**	**A**

Tabel 4. menunjukkan perbandingan karakter *Pattern* dan karakter pada *Text*. *Pattern* indeks 2 atau *pat[2]* dicocokkan dengan *Text* indeks ke 2 atau *txt[2]*. Setelah dicocokkan, diketahui bahwa kedua karakter cocok. Lalu lanjut ke indeks $i = 3$ dan $j = 3$.

Tabel 5. Mencocokkan *Pattern* ke *Text* pada *indeks* 3

Index	0	1	2	3	4	5	6	7	8	9	10
Text	**A**	**A**	**A**	**A**	**A**	**B**	**A**	**A**	**A**	**B**	**A**

Index	0	1	2	3
Text	**A**	**A**	**A**	**A**

Tabel 5 menunjukkan perbandingan karakter *Pattern* dan karakter pada *Text*. *Pattern* indeks 3 atau *pat[3]* dicocokkan dengan *Text* indeks ke 3 atau

txt[3]. Setelah dicocokkan, diketahui bahwa kedua karakter cocok. Saat *Pattern* sama dengan *Text*, maka nilai indeks disimpan ke dalam variabel *List*. Setelah itu nilai *j* direset dengan cara *j = lps[j - 1]* . Sehingga nilai *j* berubah menjadi 3, karena *lps[4-1]* = 3.

Tabel 6. Mencocokkan *Pattern* ke *Text* pada *i* = 4 dan *j* = 3

Index	0	1	2	3	4	5	6	7	8	9	10
Text	**A**	**A**	**A**	**A**	**A**	**B**	**A**	**A**	**A**	**B**	**A**

Index		0	1	2	3						
Text		**A**	**A**	**A**	**A**						

Tabel 6 menunjukkan perbandingan karakter *Pattern* dan karakter pada *Text*. *Pattern* indeks 3 atau *pat[3]* dicocokkan dengan *Text* indeks ke 4 atau *txt[4]*. Setelah dicocokkan, diketahui bahwa kedua karakter cocok. Lalu nilai *i* dan nilai *j* sama-sama dinaikkan menjadi *i* = 5 dan *j* = 4. Saat *Pattern* sama dengan *Text*, maka nilai indeks disimpan ke dalam variabel *List*. Setelah itu nilai *j* direset dengan cara *j = lps[j - 1]* . Sehingga nilai *j* berubah menjadi 3, karena *lps[4-1]* = 3.

Tabel 7. Mencocokkan *Pattern* ke *Text* pada *i* = 5 dan *j* = 3

Index	0	1	2	3	4	5	6	7	8	9	10
Text	**A**	**A**	**A**	**A**	**A**	**B**	**A**	**A**	**A**	**B**	**A**

Index			0	1	2	3					
Text			**A**	**A**	**A**	**A**					

Tabel 7 menunjukkan perbandingan karakter *Pattern* dan karakter pada *Text*. *Pattern* indeks 3 atau *pat[3]* dicocokkan dengan *Text* indeks ke 5 atau *txt[5]*. Setelah dicocokkan, diketahui bahwa kedua karakter tidak cocok.

Diketahui nilai *j* > 0, sehingga nilai *j* direset dengan cara *j = lps[j - 1]* . Sehingga nilai *j* berubah menjadi 2, karena *lps[3-1]* = 2.

Tabel 8. Mencocokkan *Pattern* ke *Text* pada *i* = 5 dan *j* = 2

Index	0	1	2	3	4	5	6	7	8	9	10
Text	**A**	**A**	**A**	**A**	**A**	**B**	**A**	**A**	**A**	**B**	**A**

Index	0	1	2	3
Text	**A**	**A**	**A**	**A**

Tabel 8 menunjukkan perbandingan karakter *Pattern* dan karakter pada *Text*. *Pattern* indeks 2 atau *pat[2]* dicocokkan dengan *Text* indeks ke 5 atau *txt[5]*. Setelah dicocokkan, diketahui bahwa kedua karakter tidak cocok. Diketahui nilai *j* > 0, sehingga nilai *j* direset dengan cara *j = lps[j - 1]* . Sehingga nilai *j* berubah menjadi 1, karena *lps[2-1]* = 1.

Tabel 9. Mencocokkan *Pattern* ke *Text* pada *i* = 5 dan *j* = 1

Index	0	1	2	3	4	5	6	7	8	9	10
Text	**A**	**A**	**A**	**A**	**A**	**B**	**A**	**A**	**A**	**B**	**A**

Index	0	1	2	3
Text	**A**	**A**	**A**	**A**

Tabel 9 menunjukkan perbandingan karakter *Pattern* dan karakter pada *Text*. *Pattern* indeks 1 atau *pat[1]* dicocokkan dengan *Text* indeks ke 5 atau *txt[5]*. Setelah dicocokkan, diketahui bahwa kedua karakter tidak cocok. Diketahui nilai *j* > 0, sehingga nilai *j* direset dengan cara *j = lps[j - 1]* . Sehingga nilai *j* berubah menjadi 0, karena *lps[1-1]* = 0.

Tabel 10. Mencocokkan *Pattern* ke *Text* pada $i = 5$ dan $j = 0$

Index	0	1	2	3	4	5	6	7	8	9	10
Text	**A**	**A**	**A**	**A**	**A**	**B**	**A**	**A**	**A**	**B**	**A**

Index	0	1	2	3
Text	**A**	**A**	**A**	**A**

Tabel 10 menunjukkan perbandingan karakter *Pattern* dan karakter pada *Text*. *Pattern* indeks 0 atau *pat[0]* dicocokkan dengan *Text* indeks ke 5 atau *txt[5]*. Setelah dicocokkan, diketahui bahwa kedua karakter tidak cocok. Diketahui nilai $j = 0$, sehingga nilai i dinaikkan menjadi $i = 6$.

Tabel 11. Mencocokkan *Pattern* ke *Text* pada $i = 6$ dan $j = 0$

Index	0	1	2	3	4	5	6	7	8	9	10
Text	**A**	**A**	**A**	**A**	**A**	**B**	**A**	**A**	**A**	**B**	**A**

Index	0	1	2	3
Text	**A**	**A**	**A**	**A**

Tabel 11 menunjukkan perbandingan karakter *Pattern* dan karakter pada *Text*. *Pattern* indeks 0 atau *pat[0]* dicocokkan dengan *Text* indeks ke 6 atau *txt[6]*. Setelah dicocokkan, diketahui bahwa kedua karakter cocok. Lalu lanjut ke indeks $i = 7$ dan $j = 1$.

Tabel 12. Mencocokkan *Pattern* ke *Text* pada $i = 7$ dan $j = 1$

Index	0	1	2	3	4	5	6	7	8	9	10
Text	**A**	**A**	**A**	**A**	**A**	**B**	**A**	**A**	**A**	**B**	**A**

Index	0	1	2	3
Text	**A**	**A**	**A**	**A**

Tabel 12 menunjukkan perbandingan karakter *Pattern* dan karakter pada *Text*. *Pattern* indeks 1 atau *pat[1]* dicocokkan dengan *Text* indeks ke 7 atau

txt[7]. Setelah dicocokkan, diketahui bahwa kedua karakter cocok. Lalu lanjut ke indeks *i* = 8 dan *j* = 2.

Tabel 13. Mencocokkan *Pattern* ke *Text* pada *i* = 8 dan *j* = 2

Index	0	1	2	3	4	5	6	7	8	9	10
Text	**A**	**A**	**A**	**A**	**A**	**B**	**A**	**A**	**A**	**B**	**A**

Index	0	1	2	3
Text	**A**	**A**	**A**	**A**

Tabel 13 menunjukkan perbandingan karakter *Pattern* dan karakter pada *Text*. *Pattern* indeks 2 atau *pat[2]* dicocokkan dengan *Text* indeks ke 8 atau *txt[8]*. Setelah dicocokkan, diketahui bahwa kedua karakter cocok. Lalu lanjut ke indeks *i* = 9 dan *j* = 3.

Tabel 14. Mencocokkan *Pattern* ke *Text* pada *i* = 9 dan *j* = 3

Index	0	1	2	3	4	5	6	7	8	9	10
Text	**A**	**A**	**A**	**A**	**A**	**B**	**A**	**A**	**A**	**B**	**A**

Index	0	1	2	3
Text	**A**	**A**	**A**	**A**

Tabel 14 menunjukkan perbandingan karakter *Pattern* dan karakter pada *Text*. *Pattern* indeks 3 atau *pat[3]* dicocokkan dengan *Text* indeks ke 9 atau *txt[9]*. Setelah dicocokkan, diketahui bahwa kedua karakter tidak cocok. Diketahui nilai *j* > 0, sehingga nilai *j* direset dengan cara *j* = *lps[j - 1]* . Sehingga nilai *j* berubah menjadi 2, karena *lps[3-1]* = 2.

Tabel 15. Mencocokkan *Pattern* ke *Text* pada $i = 9$ dan $j = 2$

Index	0	1	2	3	4	5	6	7	8	9	10
Text	**A**	**A**	**A**	**A**	**A**	**B**	**A**	**A**	**A**	**B**	**A**

Index	0	1	2	3
Text	**A**	**A**	**A**	**A**

Tabel 15 menunjukkan perbandingan karakter *Pattern* dan karakter pada *Text*. *Pattern* indeks 3 atau *pat[3]* dicocokkan dengan *Text* indeks ke 9 atau *txt[9]*. Setelah dicocokkan, diketahui bahwa kedua karakter tidak cocok. *Window* telah berada pada ujung teks, maka tak ada lagi proses pergeseran. Contoh di atas merupakan ilustrasi proses *Searching the Pattern* di dalam suatu text berdasarkan perhitungan indeks pada *Pattern* dan juga *Text* dengan melibatkan nilai *lps[]*.

F. Implementasi *Spelling Corrector*

Speling Corrector merupakan proses yang digunakan untuk melakukan koreksi kesalahan pengetikan. *Norvig Spell Corrector* merupakan metode koreksi penulisan teks yang dikembangkan oleh Peter Norvig pada tahun 2007. Metode ini menerapkan Teori Bayes untuk menghitung jumlah peluang kata yang terdapat di dalam sebuah korpus. Berikut cara kerja dari *Norvig Spell Corrector*.

1. *Selection Mechanism*

   Selection Mechanism merupakan mekanisme untuk seleksi kata yang berdasarkan pada beberapa kandidat kata dan peluang munculnya kata

dalam suatu korpus. Sehingga didapatkan kandidat dengan probabilitas gabungan tertinggi. Hal ini ditunjukkan pada Kode Program 3.

```
Algoritma Selection_Mechanism
{Mencari nilai tertinggi dari suatu probabilitas kata}

Deklarasi:
      word : string
      peluang : double

Deskripsi:
      read(word)
      read(corpus)
      P := word / count of word in corpus
      max := max value of (word, P)
write(max)
```

Kode Program 3. *Psedocode Selection Mechanism*

2. *Candidate Model*

Candidate Model adalah proses pencarian kandidat kata yang berdekatan dengan kata yang dicari. Proses tersebut dilakukan dengan *deletion* (menghapus satu huruf), *transposition* (menukar dua huruf yang berdekatan), *replacement* (mengubah satu huruf ke huruf lain), *insertion* (menambah huruf). *Pseudocode* dapat dilihat pada Kode Program 4.

```
Algoritma Candidate_Model
{Mencari pencarian kandidat kata}

Deklarasi:
     letter, word, splits, deletes, transposes, replaces,
inserts : string
Deskripsi:
     read(letters)
     splits := [(word[:i], word[i:]) for i in length of
word +1]
     deletes := [L + R[1:] for L, R in splits if R]
     transposes = [L + R[1] + R[0] + R[2:] for L, R in
splits if len(R)>1]
    replaces   = [L + c + R[1:]           for L, R in
splits if R for c in letters]
    inserts    = [L + c + R               for L, R in
splits for c in letters]
     write(set(deletes + transposes + replaces + inserts))
```

Kode Program 4. *Pseudocode Candidate Model*

3. *Language Model*

Pada proses ini, dihitung probabilitas kata dalam suatu korpus menggunakan Teori Bayes. Data di dalam korpus dipecah menjadi satu suku kata, lalu kata tersebut masing-masing dicari berapa banyak kemunculan kata dengan konsep peluang. *Pseudocode* dapat dilihat pada Kode Program 5.

```
Algoritma Language_Model
{Mencari Estimasi Peluang Kata}

Deklarasi:
     words : string
     p : double

Deskripsi:
     read(words)
     length := length of words
     P := words/length
      write(P)
```

Kode Program 5. *Pseudocode Language Model*

G. Pengujian

Pengujian yang dilakuan terdapat dua bagian. Bagian pertama untuk menguji fungsi sistem, *interface*, *database*, atau kesalahan berkaitan dengan kinerja sistem menggunakan *Black Box Testing*. Bagian kedua untuk menguji akurasi dari *Norvig Spelling Corrector* atau pengecekan kesalahan penulisan suatu kata kunci. Pengujian dilakukan untuk mencari persentase keakuratan dengan cara membandingkan hasil dari *spell corrector* dengan hasil sesungguhnya yang ingin dicapai oleh pengguna.

H. Term Frequency

Term Frequency merupakan metode untuk melakukan perhitungan bobot pada suatu kasus dalam temu kembali informasi. Frekuensi akan dihitung pada *term* dalam suatu dokumen yang bersangkutan. Semakin banyak jumlah kemunculan suatu *term* pada suatu dokumen, maka akan semakin besar bobotnya. *Term Frequency* yang digunakan adalah dengan menggunakan metode logaritmik. Logaritmik dipilih untuk menghindari dominansi dokumen yang mengandung sedikit *term* namun mempunyai frekuensi yang tinggi.

Rumus dari Term Frequency Logaritmik yaitu:

$$TF = 1 + Log10\ (Ft, d), Ftd > 0$$

atau

$$TF = 0, Ftd = 0$$

Keterangan:

TF : *Term Frequency*

Ftd : Nilai frekuensi *term* (*t*) pada dokumen (*d*)

Nilai kemunculan term bernilai nol (0) jika tak ada kemunculan pada suatu dokumen. Namun jika terdapat kemunculan *term*, maka digunakan pada rumus logaritmik tersebut.

V. SIMPULAN DAN SARAN

A. Simpulan

Simpulan yang dapat diambil berdasarkan hasil penelitian pengembangan sistem pencarian multi teks pada tujuh Kitab Hadis menggunakan Algortima *Knuth-Morris-Pratt* ialah sebagai berikut:

1. Sistem pencarian multi teks pada tujuh Kitab Hadis menggunakan Algortima *Knuth-Morris-Pratt* telah berhasil dikembangkan.
2. Sistem pencarian hadis sudah dapat mendeteksi dan memperbaiki kesalahan pengetikan pada kata kunci yang diketik oleh pengguna menggunakan *Norvig Spelling Corrector*.
3. Sistem pencarian yang dikembangkan menggunakan *framework* Django ini menghasilkan hasil pencarian Hadis yang relevan dengan kata kunci berdasarkan pengujian yang telah dilakukan.
4. Sistem pencarian hadis yang mengusung tampilan sederhana tersebut memerlukan waktu pencarian yang jauh lebih lama dibandingkan dengan sistem yang telah dikembangkan sebelumnya.
5. Sistem kompatibel di beberapa *browser* dan dapat diakses tanpa menemui kendala di dalam sistem.

B. Saran

Saran yang dapat peneliti berikan terkait dengan penelitian ini adalah sebagai berikut:

1. Mengembangkan sistem pencarian hadis menggunakan Algoritma *String Matching* yang berbeda, sehingga mendapatkan perbandingan hasil yang diperoleh dari masing-masing penelitian.
2. Mengembangkan sistem pencarian dengan menggunakan metode *Spelling Correction* yang berbeda.

DAFTAR PUSTAKA

Agustina, S. 2018. *Optimasi Pencarian Hadis Dalam Empat Kitab Hadis*. Skripsi. Lampung: Fakultas Matematika Dan Ilmu Pengetahuan Alam, Universitas Lampung.

Ambika, K.P., Ramesh, U., Saravanan, K., dan Peter, J.H. 2013. An Enhanced Version of Pattern Matching Algorithm using Bitwise XOR Operation. *International Journal of Computer Applications*. 68(23): 24-29.

APJII. 2017. *Penetrasi & Perilaku Pengguna Internet Indonesia*. Jakarta: Asosiasi Penyelenggara Jasa Internet Indonesia.

Ariani, R., dan Shalahudin, M. 2011. *Modul Pembelajaran Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Objek)*. Bandung: Modula.

Azwar, S. 2011. *Sikap manusia Teori dan Pengukurannya 2nd edition*. Yogyakarta: Pustaka Pelajar.

Broder, A. 2002. A Taxonomy of Web Search. *IBM Research*. 36(2): 5-6.

Charras, C., dan Lecroq, T. 2004. *Handbook of Exact String-Matching Algorithms*. King's College London Publications.

Connolly, T., dan Begg, C. 2005. *Database Systems A Practical Approach to Design, Implementation, and Management Fourth Edition*. Boston: Pearson Education.

Delisle, M. 2008. *Mastering phpMyAdmin 2.11 for Effective MySQL Management*. Birmingham: Packt Publishing.

Dennis, A., Wixom, B.H., dan Tegarden, D. 2012. *Systems Analysis and Design with UML 4th Edition*. John Wiley and Sons.

Dyer, R.J. 2015. *Learning MySQL and MariaDB: Heading in the Right Direction with MySQL and MariaDB*. UK: O'Reilly Media.

Hackett, C., Stonawski, M., Connor, P., dan Skirbekk, V. 2015. The Future of World Religions: Population Growth Projection. *Pew Research Center*.

Holovaty, A., Moss, J.K. 2006. *The Definitive Guide to Django: Web Development Done Right*. United States: Apress.

Islamiah, M.P. 2014. *Tata Kelola Teknologi Informasi (IT Governance) Menggunakan Framework Cobit 5: Studi Kasus Dewan Kehormatan*

Penyelenggara Pemilu (DKPP). Skripsi. Jakarta : Universitas Islam Negeri Syarif Hidayatullah.

Jogiyanto, H.M. 2005. *Analisis dan Desain Sistem Informasi: Pendekatan terstruktur teori dan praktis aplikasi bisnis*. Yogyakarta: Andi.

Kadir, A. 2014. *Pengenalan Sistem Informasi Edisi Revisi*. Yogyakarta: Andi.

Knuth, D.E., Morris, J.H., dan Pratt, V.R. 1977. Fast Pattern Matching In String. *Siam J. Comput*. 6(2): 323-349.

Kustiyahningsih, Y., dan Anamisa, D.R. 2011. *Pemrograman Basis Data Berbasisi Web Menggunakan PHP Dan MySQL*. Yogyakarta: Graha Ilmu.

Mandala, R. 2006. Evaluasi Efektifitas Metode Machine-Learning Pada Search-Engine. *Seminar Nasional Aplikasi Teknologi Informasi*.

Mandala, R., dan Waruwu, F.T. 2016. Perbandingan Algoritma Knuth-Morris-Pratt dan Boyer Moore Dalam Pencocokan String Pada Aplikasi Kamus Bahasa Nias. *Jurnal Ilmiah INFOTEK*. 1(1): 1-7.

Miller, B.N. 2008. *Java For Python Programmer*. United States: Creativecommons.

Norvig, P. 2007. *How to write a spelling corrector*. Web Page, Visited October 28th 2018, Available http://norvig.com/spell-correct.html.

Purbasari, I.Y. 2007. *Desain dan Analisis Algoritma*. Yogyakarta: Graha Ilmu.

Purwanto, E.B. 2008. *Perancangan dan Analisis Algoritma*. Yogyakarta: Graha Ilmu.

Pressman, R. 2010. *Software Engineering: A Practioner's Approach, Seventh Edition*. McGraw-Hill: New York.

Rahim, R., Zulkarnain, I., dan Jaya, H. 2017. A review: search visualization with Knuth Morris Pratt algorithm. *International Conference on Technology and Engineering*.

Sari, D.K. 2018. *Pengembangan Sistem Pencarian Pada Tujuh Kitab Hadis Menggunakan Algoritma Knuth-Morris-Pratt*. Skripsi. Lampung: Fakultas matematika Dan Ilmu Pengetahuan Alam, Universitas Lampung.

Sarno, R., Anistyasari, Y., dan Fitri, R. 2012. *Semantic Search*. Yogyakarta: Andi.

Schwab, K. 2016. *The Fourth Industrial Revolution*. Geneva: World Economic Forum.

Seymour, T., Fransvog, D., dan Kumar, S. 2011. History Of Search Engines. *International Journal of Management & Information Systems*. 15(4): 47-48.

Sidi, J.P. 2017. *Pengembangan Sistem Pencarian Informasi Pada Hadits Riwayat Bukhari*. Skripsi. Lampung: Fakultas Matematika Dan Ilmu Pengetahuan Alam, Universitas Lampung.

Sugono, D. 2008. *Kamus Bahasa Indonesia*. Jakarta: Pusat Bahasa Departemen Pendidikan Nasional.

Stackoverflow. 2018. *Developer Survey Results*. Web Page, Visited October 28th 2018, Available https://insights.stackoverflow.com/survey/2018.

Whitten, J.L., Bentley, L.D., dan Dittman, K.C. 2004. *Systems Analysis and Design Methods*. Yogyakarta: Penerbit Andi.

Yakub. 2012. *Pengantar Sistem Informasi*. Yogyakarta: Graha Ilmu.

Ziyadurrahman, R. 2018. *Pengembangan Sistem Perbandingan Pencarian Hadis Dengan Dan Tanpa Index Pada Sistem Pencarian Hadis Riwayat Imam Malik Dan Bukhari*. Skripsi. Lampung: Fakultas Matematika Dan Ilmu Pengetahuan Alam, Universitas Lampung.

Zen, E.S., dan Khairiyah, N.. 2014. *Pendidikan Agama Islam Dan Budi Pekerti: Kelas X*. Jakarta: Kementerian Pendidikan dan Kebudayaan.

Zhu, W., Xu, H., Wang, M., dan Rong, L. 2012. *U.S. Patent No. 8,176,419*. Washington, DC: U.S. Patent and Trademark Office.