SKRIPSI

APLIKASI PEMERIKSA KESALAHAAN DOKUMEN SKRIPSI INFORMATIKA UNPAR

Marcell Trixie Alexander

NPM: 2014730003

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2019

UNDERGRADUATE THESIS

GENERAL ERROR CHECKER APPLICATION FOR INFORMATICS ENGINEERING UNPAR THESIS DOCUMENT

Marcell Trixie Alexander

NPM: 2014730003

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2019

ABSTRAK

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia»

Kata-kata kunci: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Indonesia»

ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris»

Keywords: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Inggris»

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENDAHULUAN

Pada bab pendahuluan ini dijelaskan mengenai latar belakang penulisan skripsi, rumusan masalah, tujuan penulisan skripsi, batasan masalah, metodologi penelitian, dan sistematika penulisan skripsi ini.

1.1 Latar Belakang

Skripsi merupakan karangan ilmiah yang wajib ditulis oleh mahasiswa sebagai bagian dari persyaratan akhir pendidikan akademiknya. Namun dalam penulisannya, peserta skripsi sering melakukan kesalahan kecil yang tidak dapat diabaikan. Kesalahan sering terjadi dalam penggunaan imbuhan, kata keterangan, penulisan kata dan sebagainya. Hal-hal seperti ini seharusnya dapat diperiksa dan diminimalisir oleh diri sendiri. Pada saat bimbingan, waktu dosen pembimbing lebih baik dimanfaatkan untuk membahas konten dibanding memeriksa kesalahan-kesalahan tersebut.

Dari masalah tersebut dapat dibuat sebuah aplikasi untuk melakukan pemeriksaan pada dokumen skripsi. Kesalahan yang akan diperiksa berasal dari survei yang dilakukan kepada dosen-dosen Informatika Unpar. Hasil dari survei tersebut akan diseleksi untuk diimplementasikan ke dalam aplikasi. Aplikasi sederhana ini dapat dimanfaatkan oleh mahasiswa Informatika Unpar secara mandiri. Aplikasi ini dijalankan melalui melalui terminal *command Windows*. Aplikasi menerima masukan berupa file *PDF* skripsi dan menampilkan laporan yang berisi kesalahan-kesalahan yang ditemukan pada dokumen skripsi.

1.2 Rumusan Masalah

Berdasarkan deskripsi topik yang sudah ditulis, dapat dirumuskan masalah sebagai berikut:

1. Bagaimana cara memeriksa kesalahan yang ada pada dokumen skripsi?
2. Bagaimana cara membuat perangkat lunak yang dapat memeriksa kesalahan pada dokumen skripsi?

1.3 Tujuan

Tujuan dari skripsi ini adalah sebagai berikut:

1. Dapat memeriksa kesalahan yang ada pada dokumen skripsi

2. Dapat membangun perangkat lunak untuk memeriksa kesalahan yang ada pada dokumen skripsi

1.4 Batasan Masalah

Batasan masalah skripsi ini adalah sebagai berikut:

1. Pemeriksaan menggunakan *pattern matching* tanpa analisis gramatikal
2. Tidak ada pemeriksaan kosakata

1.5 Metodologi Penelitian

Metodologi penelitian yang digunakan pada skripsi ini adalah sebagai berikut:

1. Melakukan survei kepada dosen-dosen Informatika mengenai kesalahan-kesalahan penulisan yang ditemui dalam dokumen skripsi
2. Melakukan studi literatur *Regular Expression* untuk mendeteksi kesalahan-kesalahan dalam file *PDF* skripsi
3. Mempelajari *library PDF Parser* untuk mengestraksi file *PDF* skripsi yang akan diperiksa
4. Melakukan perancangan perangkat lunak
5. Melakukan implementasi perancangan perangkat lunak
6. Melakukan pengujian terhadap perancangan perangkat lunak
7. Menulis dokumen skripsi

1.6 Sistematika Pembahasan

Sistematika penulisan pada skripsi ini terdiri dari 6 bab, yaitu:

1. Bab 1 Pendahuluan
   Bab 1 akan membahas latar belakang dibuatnya perangkat lunak untuk memeriksa kesalahan dokumen skripsi. Pada bab ini dibahas juga rumusan masalah, tujuan skripsi, batasan masalah dan metodologi penelitian yang digunakan pada skripsi.
2. Bab 2 Landasan Teori
   Bab 2 yang merupakan landasan teori akan berisi teori-teori yang menjadi dasar-dasar dalam penulisan skripsi ini. Teori yang akan dibahas pada bab 2, yaitu *Regular Expression* dan *library PDF Parser*.
3. Bab 3 Analisis Masalah
   Bab 3 berisi analisis masalah yang muncul dalam menyelesaikan masalah tersebut. Pada bab ini akan dianalisa masalah yang ditemukan pada saat melakukan pengamatan beberapa sidang

skripsi semester Ganjil 2018/2019 dan survei secara personal kepada dosen-dosen Informatika Unpar. Selain itu, akan dibahas mengenai hal-hal yang dibutuhkan oleh perangkat lunak yang akan dibuat.

4. Bab 4 Perancangan
   Bab 4 berisi rancangan perangkat lunak yang akan dibuat. Perangkat lunak akan dibangun menggunakan bahasa pemrograman *PHP*.

5. Bab 5 Implementasi dan Pengujian
   Bab 5 pada skripsi ini membahas implementasi perangkat lunak dan pengujian yang dilakukan terhadap perangkat lunak tersebut. Bab ini juga menjelaskan tentang spesifikasi perangkat lunak dan pengujian yang dilakukan pada skripsi ini.

6. Bab 6 Kesimpulan dan Saran
   Bab 6 berisi kesimpulan dari penulisan skripsi ini. Bab ini juga berisi saran untuk pengembangan perangkat lunak agar lebih baik lagi.

BAB 2

LANDASAN TEORI

Pada bab 2 akan diuraikan tentang landasan teori yang membahas *regular expression* dan *library PDF Parser*.

2.1 *Regular Expression*

Regular expression (*regex*) [1] adalah jenis pola teks tertentu yang dapat digunakan pada banyak aplikasi modern dan bahasa pemrograman. *Regex* biasanya dimanfaatkan untuk memverifikasi kecocokan antara input dengan pola teks, untuk menemukan teks yang cocok dengan pola dalam teks yang lebih besar, untuk mengganti teks yang cocok dengan pola dengan teks lain atau menyusun ulang bit dari teks yang cocok dan untuk membagi sebuah blok teks menjadi beberapa subteks. *Regex*sudah banyak digunakan dalam pencocokan pola, misalnya untuk validasi beberapa string seperti *username* dan *password*, alamat *e-mail*, alamat *IP* ataupun nomor telepon. Pemanfaatan *regex* dengan baik, dapat menyederhanakan banyak tugas pemrograman dan pemrosesan teks dalam kehidupan sehari-hari.

Istilah *regex* berasal dari teori matematika dan komputer sains, yang mencerminkan sifat ekspresi dalam matematika yang disebut keteraturan. Ekspresi tersebut dapat diimplementasikan dalam perangkat lunak, dengan menggunakan *Deterministic Finite Automaton* (DFA). DFA adalah *finite state machine* yang tidak menggunakan *backtracking*.

Regex dapat digunakan dalam berbagai bahasa pemrograman, salah satunya yaitu, *Perl Compatible Regular Expression* (PCRE). PCRE [2] adalah serangkaian fungsi yang menerapkan pencocokan pola *regex* dengan menggunakan sintaks dan semantik yang sama dengan bahasa pemrograman *Perl 5*, meskipun ada beberapa sedikit perbedaan. Pada saat ini, implementasi yang digunakan sesuai dengan *Perl* versi 5.005.

2.1.1 Metakarakter

Metakarakter pada *regex* dibedakan menjadi 2 jenis berdasarkan dari posisinya, yaitu metakarakter *outside square brackets* dan metakarakter *outside square brackets*. Meskipun ada beberapa simbol metakarakter yang sama, namun fungsinya agak berbeda. Pada metakarakter *outside square brackets* terdapat 14 simbol, sedangkan metakarakter *inside square brackets* terdapat 3 simbol. Rincian dari ke-2 metakarakter tersebut akan dijelaskan sebagai berikut:

Tabel 2.1: Tabel metakarakter *outside square brackets*

Simbol	Nama
\	Backslash
^	Circumflex Anchor
$	Dollar Anchor
.	Dot
[	Left Square Bracket
]	Right Square Bracket
|	Vertical Bar
(	Left Parenthesis
)	Right Parenthesis
?	Question Mark
*	Asterisk
+	Plus
{	Left Curly Brace
}	Right Curly Brace

Pada tabel 2.1 telah disebutkan simbol-simbol yang digunakan pada metakarakter *outside square brackets*. Berikut ini adalah penjelasan fungsi dari setiap metakarakter:

1. *Backslash*
   Backslash yang berada di luar kelas karakter memiliki beberapa fungsi, yaitu:
   - Membuat karakter lepas
     Apabila diikuti oleh karakter non-alfanumerik, metakarakter ini dapat menghilangkan makna khusus yang dimiliki oleh karakter tersebut. Misalnya pada saat ingin mencocokan karakater "*", dengan menggunakan metakarkter *backslash* dapat dituliskan dengan "*". Jadi karakter *asterisk* akan terbaca sebagai karakter bukan sebagai metakarakter.
   - Menggunakan karakter yang tidak dapat ditulis dalam pola, seperti \a (*alarm*), \cx (*control-x*, \e (*escape*) dan lain-lain.
   - Menentukan jenis karakter dalam pola, seperti \d (angka desimal), \D (non-angka desimal), \w (karakter kata), \W (non-karakter kata) dan lain-lain.
   - Menggunakan pernyataan sederhana tertentu, seperti \b (*word boundary*), \B (*non-word boundary* dan lain-lain.
2. *Anchor*
   Anchor merupakan metakarakter yang terdiri dari simbol *circumflex* (^) dan *dollar* ($). *Circumflex* digunakan sebagai tanda awal yang memulai suatu teks string, sedangkan *dollar* digunakan sebagai tanda akhir dari suatu teks pola.
3. *Dot*
   Dot akan cocok dengan karakter apapun, keculai karakter *newline*.
4. *Square brackets*
   Square brackets terdiri dari pasangan "[" dan "]", memiliki fungsi untuk mendefinisikan kelas

karakter. Kelas karakter akan berada di antara 2 karakter tersebut. Contohnya kelas karakter numerik [0-9] sama dengan [0123456789].

5. *Vertical bar*
   Vertical bar berfungsi untuk memisahkan beberapa kondisi pola alternatif yang berbeda. Sebagai contohnya untuk pola A | B, maka hasilnya akan mencocokan ”A” atau ”B”.

6. *Parenthesis*
   Parenthesis terdiri dari pasangan ”(” dan ”)”, memiliki fungsi untuk mengelompokan subpola dalam *regex*.

7. *Quantifiers*
   Quantifiers berfungsi untuk menunjukkan berapa banyak instance karakter, set karakter, atau kelas karakter yang harus dicocokkan. Pada metakarakter ini terdapat 3 jenis, yaitu:

   - *question mark* (?), dengan jumlah kuantifier antara 0 hingga 1 0,1.
   - *asterisk* (*), dengan jumlah kuantifier 0 atau lebih 0,.
   - *plus* (+), dengan jumlah kuantifier 1 atau lebih 1,.

8. *Curly Brackets*
   Curly Brackets terdiri dari pasangan ”” dan ””, memiliki fungsi untuk mendefinisikan angka minimal dan maksimum dari kuantifiers yang akan digunakan. Misalnya pola a1,5, akan cocok dengan ”a”, ”aa”, ”aaa”, ”aaaa” atau ”aaaaa”.

Metakarakter *inside square brackets* merupakan bagian dari kelas karakter. Pada bagian kelas karakter ini hanya terdapat 3 macam metakarakter saja. Berikut adalah metakarakter yang digunakan:

Tabel 2.2: Tabel metakarakter *inside square brackets*

Simbol	Nama
\	Backslash
^	Circumflex
-	Hyphen

Pada tabel 2.2 telah disebutkan macam-macam metakarakter *inside square brackets*. Dari ke-3 metakarakter tersebut ada beberapa yang memiliki simbol yang sama dengan metakarakter *outside square brackets*, namun fungsinya berbeda. Berikut adalah penjelasan dari fungsi metakarakter dari tabel 2.2:

1. *Backslash*
   Metakarakter ini fungsinya sama dengan *backslash* yang ada pada *outside square brackets*. Namun dari ke-4 fungsi tersebut hanya 3 fungsi saja yang digunakan, yaitu membuat karakter lepas, Menggunakan karakter yang tidak dapat ditulis dalam pola dan menentukan jenis karakter dalam pola.

2. *Circumflex*
   Metakarakter ini memiliki fungsi yang berbeda dengan yang digunakan pada *outside square brackets*. Fungsinya untuk membuat negasi, namun hanya berlaku untuk karakter pertamanya saja. Contohnya [^0] akan cocok dengan semua karakter kecuali karakter 0.

3. *Hyphen*
   Metakarakter ini berfungsi untuk menentukan jangkauan dari sebuah karakter dalam kelas karakter, seperti [0-9] yang menandakan jangkauan karakter dari angka 0 hingga 9.

2.1.2 Kelas Karakter

Kelas karakter adalah karakter yang memiliki atribut yang spesifik yang dikelompokan dalam sebuah kelas. Karakter tersebut dapat berbeda di setiap negara. Kelas karakter hanya valid digunakan pada *regex* didalam tanda kurung siku pada *bracket expression*. Perl mendukung notasi POSIX yang digunakan untuk kelas karakter. Dalam penggunaannya, kelas-kelas tersebut ditulis diantara "[:" dan ":]". PCRE juga mendukung penggunaan notasi ini. Sebagai contoh untuk kelas alfanumerik, penulisannya yaitu [:alnum:]. Berikut ini akan dijelaskan macam-macam kelas karakter yang digunakan:

Tabel 2.3: Tabel kelas karakter

Kelas	Deskripsi
alnum	Alfanumerik
alpha	Huruf
ascii	Kode karakter (0-127)
blank	Spasi dan Tab
cntrl	Karakter kontrol
digit	Angka desimal
graph	Karakter cetak (kecuali spasi)
lower	Huruf kecil
print	Karakter cetak (termasuk spasi)
punct	Karakter cetak (kecuali alfanumerik)
space	*White space*
upper	Huruf kapital
word	Karakter "word"
xdigit	Heksadesimal

Pada tabel 2.3, telah diuraikan 14 jenis kelas karakter. Masing-masing dari kelas karakter tersebut, akan dijelaskan sebagai berikut:

1. *alnum*
   Kelas yang berisi dengan karakter alfanumerik, meliputi angka dan huruf. Karakter-karakter yang termasuk yaitu, huruf a-Z atau A-Z dan angka 0-9. Simbol atau karakter khusus tidak termasuk dalam kelas ini.

2. *alpha*
   Kelas yang berisi dengan karakter huruf. Karakter-karakter yang termasuk yaitu, huruf a-Z

atau A-Z. Simbol atau karakter khusus tidak termasuk dalam kelas ini.

3. *ascii*
   Kelas yang merepresentasikan kode karakter dari 0-127.

4. *blank*
   Karakter-karakter yang termasuk yaitu, TAB dan spasi.

5. *control*
   Karakter kontrol adalah karakter yang tidak merepresentasikan simbol tetapi merepresentasikan character encoding.

6. *digit*
   Kelas yang berisi dengan karakter numerik. Karakter-karakter yang termasuk yaitu, angka 0-9.

7. *graph*
   Kelas yang berisi karakter yang dapat dicetak dan tampak. Contoh karakter yang dapat dicetak namun tidak tampak adalah karakter spasi dan TAB. Jadi pada kelas ini karakter spasi dan tab tidak termasuk.

8. *lower*
   Kelas yang berisi karakter dengan huruf kecil.

9. *print*
   Kelas yang berisi karakter yang dapat dicetak. Karakter yang termasuk kelas ini adalah spasi.

10. *punct*
    Kelas yang berisi karakter yang dapat dicetak, namun tidak termasuk alfanumerik.

11. *space*
    Karakter yang termasuk kelas ini adalah spasi dan TAB.

12. *upper*
    Kelas yang berisi karakter dengan huruf kapital.

13. *word*
    Kelas yang berisi karakter kata.

14. *xdijit*
    Kelas yang merepresentasikan bilangan hexadesimal, a-f atau A-F dan 0-9.

2.2 PDF Parser

PDF Parser [3] adalah sebuah *PHP library* yang digunakan untuk mengekstrak data yang ada dalam sebuah file PDF. Fitur-fitur yang sudah ada dalam *library* ini, yaitu:

1. Memuat / mengurai objek dan header

2. Menampilkan data meta, seperti nama penulis, deskripsi dan sebagainya
3. Menampilkan isi dari teks PDF
4. Dapat digunakan untuk kompresi PDF
5. Mendukung *MAC OS Roman charset encoding*
6. Menangani *encoding* heksa dan oktal pada teks

Dari fitur yang sudah ada, ada beberapa yang masih belum bisa ditangani oleh *library* ini. *PDF Parser* belum dapat mengekstrak dokumen yang diamankan. Selain itu, *library* ini tidak dapat mendeteksi jenis teks yang dicetak miring, tebal dan bergaris bawah. Hingga saat ini, pengembangan *library PDF parser* masih terus berjalan.

2.3 Kamus Indonesia *LibreOffice*

LibreOffice adalah sebuah paket aplikasi perkantoran berfitur lengkap yang tersedia secara gratis. LibreOffice menggunakan *Open Document Format* (ODF) sebagai format aslinya untuk menyimpan dokumen. ODF menjadi format standar terbuka yang sedang diadopsi sebagai format file yang dibutuhkan untuk penerbitan dan penerimaan dokumen. LibreOffice juga dapat membuka dan menyimpan dokumen dalam format lainnya. Salah satunya format yang digunakan oleh beberapa versi dari Microsoft Office.

LibreOffice memiliki ekstensi untuk kamus Indonesia. Ekstensi ini sudah mengalami beberapa perkembangan, mulai dari versi 1.0 yang rilis pada tanggal 19 Mei 2012. Versi 1.0 merupakan hasil unggahan kembali dari ekstensi *OpenOffice* yang terakhir diperbaharui pada tahun 2009. Pada tanggal 17 Mei 2014 versi 1.1 dirilis, pada versi ini LibreOffice versi 4.0 dapat menggunakan ekstensi ini. Selanjutnya, pada tanggal 15 Juli 2014 versi 2.0 dirilis. Pada versi yang paling baru ini digunakan metode baru dengan sirkumfiks yang kemudian mengubah daftar kata, sehingga memuat semua lema dari Kamus Besar Indonesia.

BAB 3

ANALISIS MASALAH

Pada bab 3 akan diuraikan tentang survei kesalahan umum.

3.1 Survei Kesalahan Umum

Pada subbab ini, akan diuraikan tentang survei yang dilakukan untuk mengumpulkan informasi yang dibutuhkan dalam pengembangan perangkat lunak. Informasi yang dicari adalah tentang kesalahan-kesalahan umum yang sering terjadi pada penulisan dokumen skripsi. Untuk mengumpulkan informasi tersebut, metode yang dipilih adalah melakukan survei. Dalam pelaksanaannya, survei dibagi menjadi dua, yaitu pengamatan beberapa sidang skripsi dan wawancara secara personal kepada dosen-dosen Informatika Unpar.

3.1.1 Pengamatan Sidang

Pengamatan dilakukan pada sidang skripsi semester Ganjil 2018/2019, yang berlangsung pada bulan Mei 2019. Tidak semua sidang skripsi yang berlangsung diamati, melainkan dari 42 sidang skripsi hanya diambil 7 sidang skripsi saja. Hal tersebut dilakukan dengan pertimbangan dari ke-7 sidang skripsi tersebut diuji oleh dosen Informatika yang berbeda-beda. Namun ada beberapa dosen Informatika yang tidak masuk dalam pengamatan, karena tidak dapat menghadiri sidang yang diuji oleh dosen tersebut. Berikut adalah rincian dari sidang skripsi yang telah diamati:

1. Pengamatan tanggal 15 Mei 2019
   Sidang yang diamati adalah sidang skripsi Osfaldo Mickael Oktavianus Naibaho, dengan judul Sistem Informasi Penjualan Barang Pada Apotek. Sidang tersebut diuji oleh Vania Natali, S.Kom, M.T. dan Elisati Hulu, M.T..

2. Pengamatan tanggal 16 Mei 2019
   Sidang yang diamati adalah sidang skripsi Ricky Wahyudi, dengan judul Temu Kembali Gambar Menggunakan Fitur Surf dan Warna. Sidang tersebut diuji oleh Vania Natali, S.Kom, M.T. dan Elisati Hulu, M.T..Dr.rer.nat. Cecilia Esti Nugraheni, ST, MT dan Dr. Ir. Veronica Sri Moertini, MT.

3. Pengamatan tanggal 17 Mei 2019
   Sidang yang diamati adalah sidang skripsi Billy Adiwijaya, dengan judul Pembangkit Timelapse Pengembangan Proyek Perangkat Lunak. Sidang tersebut diuji oleh Kristopher David Harjono M.T. dan Elisati Hulu M.T..

4. Pengamatan tanggal 20 Mei 2019
   Sidang yang diamati adalah sidang skripsi Ihsan Fajari, dengan judul Sistem Informasi Rekomendasi Pariwisata di Tasikmalaya. Sidang tersebut diuji oleh Dra. Rosa de Lima Endang Padmowati, MT dan Dr. Ir. Veronica Sri Moertini, MT.

5. Pengamatan tanggal 22 Mei 2019
   Sidang yang diamati adalah sidang skripsi Muhammad Adrian Putra Zubir, dengan judul Sistem Informasi Penyediaan Barang Pada Apotek. Sidang tersebut diuji oleh Dra. Rosa de Lima Endang Padmowati, MT dan Pascal Alfian Nugroho, S.Kom, M.Comp.

6. Pengamatan tanggal 23 Mei 2019
   Sidang yang diamati adalah sidang skripsi Ellena Angelica, dengan judul Kolektor Pengumuman Informatika. Sidang tersebut diuji oleh Natalia S.Si, M.Si dan Dr. Ir. Veronica Sri Moertini, MT.

7. Pengamatan tanggal 24 Mei 2019
   Sidang yang diamati adalah sidang skripsi Evelyn Wijaya, dengan judul Open Source Snake 360. Sidang tersebut diuji oleh Candra Wijaya S.T., M.T. dan Raymond Chandra Putra, S.T., M.T..

Dari ke-7 pengamatan yang telah dilakukan, terdapat beberapa kesalahan-kesalahan yang terjadi dalam penulisan dokumen skripsi. Berikut adalah kesalahan-kesalahan yang disebutkan oleh dosen penguji pada sidang skripsi di atas beserta penjelasannya:

1. Penulisan judul
   Kode implementasi: PS-01
   Dalam penulisan judul, setiap huruf awal pada kata harus menggunakan huruf kapital. Hal ini berlaku untuk hampir semua jenis kata, seperti nama, tempat, sifat dan keterangan. Namun, ada beberapa pengecualian seperti preposisi (kata depan yang diikuti oleh kata lainnya), konjungsi (kata sambung), dan interjeksi (kata yang mengungkapkan seruan perasaan).

2. Penulisan kata
   Kode implementasi: PS-02
   Mahasiswa paling sering melakukan kesalahan dalam penulisan kata, atau yang lebih sering disebut dengan *typo*. Kesalahan-kesalahan kecil seperti ini paling sering terjadi tanpa disadari.

3. Penggunaan imbuhan di- dan kata depan di
   Kode implementasi: PS-03
   Kesalahan ini merupakan kesalahan yang sering terjadi dalam penulisan dokumen skripsi. Penulisan imbuhan di- disatukan antara imbuhan dengan kata dasarnya. Untuk kata depan, penulisannya dipisah antara kata depan dengan kata berikutnya. Pada umumnya diikuti oleh keterangan tempat atau waktu.

4. Pemberian spasi sebelum dan setelah tanda baca
   Kode implementasi: PS-04
   Salah satu hal kecil yang sering mengganggu adalah penggunaan spasi sebelum dan setelah

tanda baca. Tanda baca yang paling sering dipakai, seperti titik, koma, tanya, dan seru harus diberi spasi setelahnya. Spasi juga digunakan sebelum menggunakan tanda kurung buka. Ada beberapa kesalahan yang masih ditemukan seperti, memberi spasi sebelum tanda tanya ataupun memberi spasi sebelum dan setelah garis miring.

5. Awal kalimat tidak menggunakan huruf kapital
   Kode implementasi: PS-05
   Setiap huruf pertama pada kata pertama dalam sebuah kalimat harus ditulis dengan huruf kapital.

6. Terdapat ruang kosong yang besar
   Kode implementasi: PS-06
   Masalah ini sering ditemukan dalam penulisan dokumen skripsi, biasanya terjadi pada saat menyisipkan gambar atau tabel. Susunan atau ukuran gambar yang tidak tepat dapat mengakibatkan terciptanya ruang kosong yang besar.

7. Tidak ada spasi antar kata
   Kode implementasi: PS-07
   Setiap kata dalam sebuah kalimat dipisahkan dengan jarak 1 spasi agar kalimat dapat dibaca dan dimengerti dengan baik.

8. Gambar tidak sesuai tempatnya
   Kode implementasi: PS-08
   Pada PDF Latex, biasanya kesalahan ini karena mahasiswa tidak memberikan tag kepada gambar tersebut. Hal ini mengakibatkan posisi gambar tidak terletak pada tempat yang seharusnya.

9. Tidak ada keterangan untuk gambar dan tabel
   Kode implementasi: PS-09
   Dalam penulisan dokumen skripsi, setiap gambar dan tabel perlu diberikan keterangan.

10. Jumlah subbab dalam 1 bab tidak boleh hanya 1
    Kode implementasi: PS-10
    Dalam sebuah bab, biasanya jumlah subbab lebih dari 1. Kesalahan yang sering dilakukan oleh mahasiswa yaitu, hanya terdapat 1 subbab saja pada 1 bab. Apabila dalam bab tersebut hanya terdapat 1 subbab, lebih baik tidak perlu dibuat subbab.

3.1.2 Wawancara Personal

Survei dilanjutkan dengan melakukan wawancara secara personal kepada dosen-dosen Informatika Unpar. Berikut adalah hasil wawancara yang sudah dilakukan:

1. Wawancara pertama dilakukan pada tanggal 9 Juli 2019. Dosen yang menjadi narasumbernya adalah Keenan Adiwijaya Leeman S.T.. Kesalahan penulisan dokumen skripsi yand didapat adalah sebagai berikut:

- Cetak miring untuk bahasa asing
  Kode implementasi: KAL-01
  Penggunaan kata dalam bahasa asing harus ditulis menggunakan cetak miring. Mahasiswa sering lupa untuk menulis cetak miring bahasa asing.
- Kalimat pengantar untuk setiap subbab
  Kode implementasi: KAL-02
  Setiap penulisan bab dan subbab selalu diikuti dengan kalimat pengantar untuk memulai bab dan subbab tersebut. Kesalahan yang sering terjadi, yaitu mahasiswa seringkali lupa untuk menuliskan kalimat pengantar tersebut.
- Kelengkapan data skripsi
  Kode implementasi: KAL-03
  Data skripsi harus diisi dengan lengkap sebagai bentuk identitas, seperti nama mahasiswa, NPM, dosen pembimbing, judul skripsi dan sebagainya. Hal-hal seperti seringkali lupa diisi karena terlalu fokus dalam mengerjakan konten-konten dalam skripsi.

2. Wawancara ke-2 dilakukan pada tanggal 9 Juli 2019. Dosen yang menjadi narasumbernya adalah Candra Wijaya S.T., M.T.. Kesalahan penulisan dokumen skripsi yand didapat adalah sebagai berikut:

   - Letak keterangan untuk gambar dan tabel
     Kode implementasi: CHW-01
     Kesalahan yang sering terjadi adalah letak dari penulisan keterangan tersebut. Keterangan pada gambar posisinya ada di bawah gambar, sedangkan keterangan pada tabel posisinya ada di atas tabel.
   - Penggunaan bahasa yang benar
     Kode implementasi: CHW-02
     KBBI menjadi kaidah dalam penulisan bahasa Indonesia. Mahasiswa terkadang salah memilih kata yang hendak ditulis dalam dokumen, padahal kata tersebut tidak sesuai dengan KBBI.

3. Wawancara ke-3 dilakukan pada tanggal 15 Juli 2019. Dosen yang menjadi narasumbernya adalah Husnul Hakim, S.Kom., M.T.. Kesalahan penulisan dokumen skripsi yand didapat adalah sebagai berikut:

   - Rujukan untuk gambar dan tabel
     Kode implementasi: HUH-01
     Setiap gambar dan tabel yang dimasukan ke dalam dokumen skripsi, perlu dirujuk dalam sebuah paragraf. Mahasiswa sering lupa atau terlewat untuk merujuk gambar dan tabel tersebut.
   - Penulisan pseudocode
     Kode implementasi: HUH-02
     Dalam penulisan pseudocode hal-hal yang perlu diperhatikan antara lain nama method, masukan serta keluaran pada method dan no baris pada pseudocode.

- Penulisan kata hubung
  Kode implementasi: HUH-03
  Kesalahaan penggunaan konjungsi akan berakibat tidak jelasnya makna kalimat karena hubungan antar frasa dan antar klausa tidak jelas.

4. Wawancara ke-4 dilakukan pada tanggal 16 Juli 2019. Dosen yang menjadi narasumbernya adalah Vania Natali, S.Kom, M.T. Kesalahan penulisan dokumen skripsi yand didapat adalah sebagai berikut:

   - Tahun skripsi pada cover skripsi
     Kode implementasi: VAN-01
     Penulisan tahun skripsi harus sama dengan tahun dimana mahasiswa mengambil skripsi tersebut. Kesalahan yang pernah terjadi, yaitu mahasiswa salah menuliskan tahun skripsi. Meskipun terlihat sepele, namun hal ini perlu diperhatikan.
   - Konsistensi penggunaan kata
     Kode implementasi: VAN-02
     Mahasiswa harus konsisten dalam penulisan kata, misalnya kata*user* dan pengguna. Mahasiswa harus memilih antara memakai *user* atau pengguna.
   - Penggunaan kata ganti orang
     Kode implementasi: VAN-03
     Dalam penulisan dokumen skripsi, tidak boleh ada kata ganti orang. Jika karya non-ilmiah lebih santai karena memakai gaya bahasa non-formal, maka berbeda dengan karya ilmiah. Karya ilmiah memiliki aturan baku dan menggunakan bahasa formal.

5. Wawancara ke-5 dilakukan pada tanggal 16 Juli 2019.

   - Penulisan daftar referensi
     Kode implementasi: NAT-01
     Penulisan daftar referensi dibuat jika dalam tulisan ilmiah tersebut memang menggunakan kutipan atau rujukan dari orang lain. Kesalahan yang sering terjadi, yaitu tidak ditemukannya referensi yang akan digunakan. Pada teks yang akan dirujuk, akan terdapat tanda [?], seharusnya tanda tanya tersebut diisi oleh nomor dari referensi.

3.1.3 Keputusan Implementasi Hasil Survei

Setiap hasil survei yang didapatkan melalui pengamatan sidang skripsi dan wawancara dosen, telah diberikan sebuah kode. Kode tersebut akan digunakan sebagai identitas dari setiap hasil survei. Namun, tidak semua dari hasil survei tersebut dapat diimplementasikan menggunakan *regex*. Metode yang digunakan untuk mendeteksi kesalahan yaitu dengan *pattern matching*, sehingga hal-hal yang bersifat kontekstual tidak dapat dicek dengan *regex*. Berikut ini adalah hasil keputusan yang telah diambil pada setiap hasil survei di atas:

Tabel 3.1: Tabel keputusan implementasi

Kode	Keputusan	
Survei	Bisa Diimplentasi	Tidak Bisa
PS-01	✓	-
PS-02	-	✓
PS-03		
PS-04	✓	-
PS-05	✓	-
PS-06	-	✓
PS-07	✓	-
PS-08	-	✓
PS-09	-	✓
PS-10	✓	-
KAL-01	-	✓
KAL-02		
KAL-03	✓	-
CHW-01	-	✓
CHW-02	✓	-
HUH-01		
HUH-02	-	✓
HUH-03		
VAN-01	✓	-
VAN-02		
VAN-03	✓	-
NAT-01	✓	-

DAFTAR REFERENSI

[1] Goyvaerts, J. dan Levithan, S. (2012) *Regular Expressions Cookbook*, 2nd edition. O?Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

[2] PHP Perl compatible regular expression. https://www.php.net/manual/en/book.pcre.php. 24 Juli 2019.

[3] Parser, P. Pdf parser. https://www.pdfparser.org/. 7 Mei 2019.

LAMPIRAN A

KODE PROGRAM

Listing A.1: MyCode.c

```c

// This does not make algorithmic sense,
// but it shows off significant programming characters.

#include<stdio.h>

void myFunction( int input, float* output ) {
    switch ( array[i] ) {
        case 1: // This is silly code
            if ( a >= 0 || b <= 3 && c != x )
                *output += 0.005 + 20050;
            char = 'g';
            b = 2^n + ~right_size - leftSize * MAX_SIZE;
            c = (--aaa + &daa) / (bbb++ - ccc % 2 );
            strcpy(a,"hello $@?");
    }
    count = ~mask | 0x00FF00AA;
}

// Fonts for Displaying Program Code in LATEX
// Adrian P. Robson, nepsweb.co.uk
// 8 October 2012
// http://nepsweb.co.uk/docs/progfonts.pdf
```

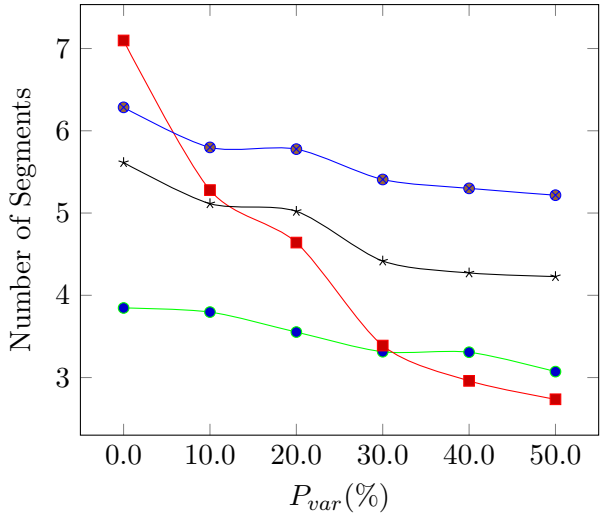
Listing A.2: MyCode.java

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;

//class for set of vertices close to furthest edge
public class MyFurSet {
    protected int id;                                   //id of the set
    protected MyEdge FurthestEdge;                      //the furthest edge
    protected HashSet<MyVertex> set;                    //set of vertices close to furthest edge
    protected ArrayList<ArrayList<Integer>> ordered;    //list of all vertices in the set for each trajectory
    protected ArrayList<Integer> closeID;               //store the ID of all vertices
    protected ArrayList<Double> closeDist;              //store the distance of all vertices
    protected int totaltrj;                             //total trajectories in the set

    /*
     * Constructor
     * @param id : id of the set
     * @param totaltrj : total number of trajectories in the set
     * @param FurthestEdge : the furthest edge
     */
    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
        this.id = id;
        this.totaltrj = totaltrj;
        this.FurthestEdge = FurthestEdge;
        set = new HashSet<MyVertex>();
        ordered = new ArrayList<ArrayList<Integer>>();
        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
        closeID = new ArrayList<Integer>(totaltrj);
        closeDist = new ArrayList<Double>(totaltrj);
        for (int i = 0;i <totaltrj;i++) {
            closeID.add(-1);
            closeDist.add(Double.MAX_VALUE);
        }
    }

}
```

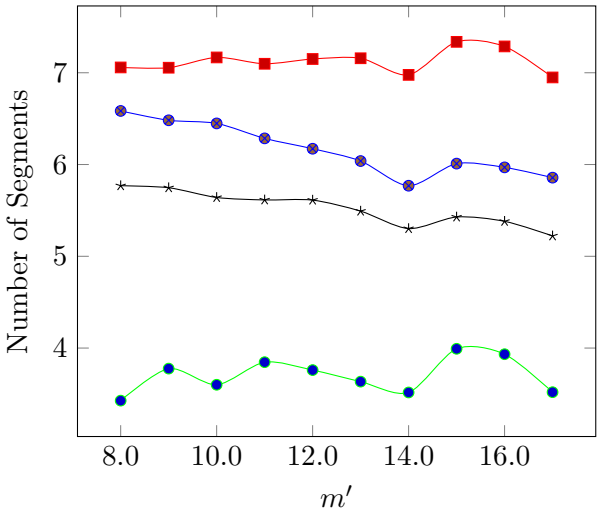
LAMPIRAN B

HASIL EKSPERIMEN

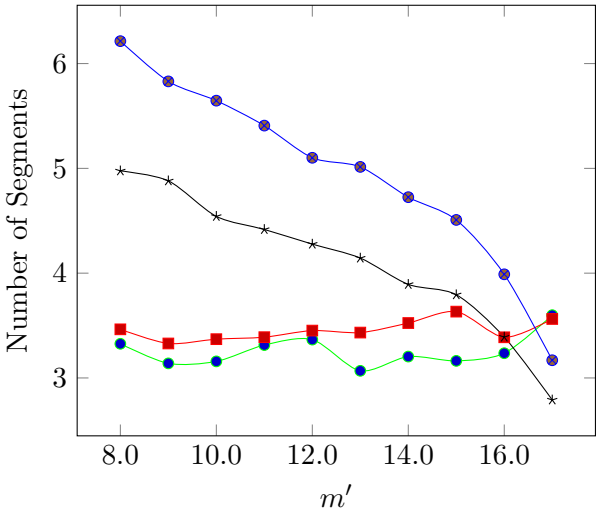
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



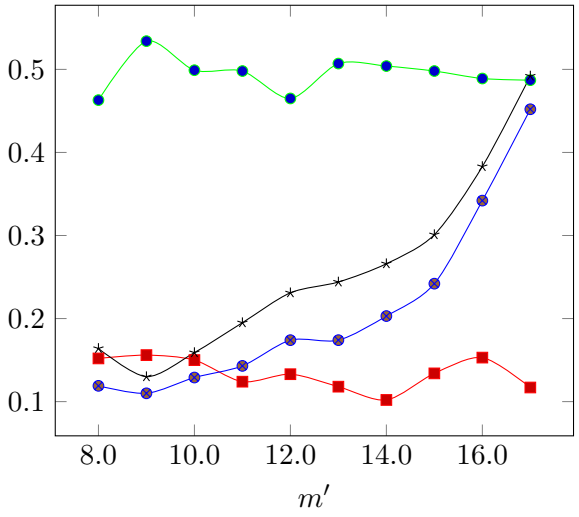
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4